

Project Report
*For the BSc computer science Final year project
at City, University of London
Academic year 2021/2022*

Project title:

Visual Fabric Classification using Neural Networks

Student: Nathan Odibo | Email: nathan.odibo@city.ac.uk

Consultant: Dr. Martin Walter

Proposed by: Nathan Odibo

INDEX

Project Report.....	1
Abstract	5
Chapter 1: Introduction	6
1.1 Problem solved	6
1.2 Project Objectives.....	6
1.2.1 Main Objective	6
1.2.2 Additional Objectives.....	6
1.2.3 Optional Objectives	7
1.3 Beneficiaries	7
1.4 Work Performed	7
1.5 Assumptions made	7
Chapter 2: Output Summary	8
2.1 The fabric datasets	8
2.2 Model files	8
2.3 The fabric classification app(s)	8
2.4 Utility Files	9
Chapter 3: Literature Review.....	9
Machine Learning	9
Deep Learning.....	10
Convolutional Neural Networks (CNNs)	10
Similar Work	10
Data expansion through augmentation and Fabric identification	10
Creating and adjusting a suitable Model.....	11
L1 Regularisation	11
Chapter 4: Methods.....	12
Software development processes	12
Management Tools.....	12
Data Acquisition.....	12
Model Adjustments	12
Model Architecture	14
System setup	14
4.1. Dataset Preparation	15
4.2 Model Creation.....	15
4.2.1 Model 1: Testing 2-Layer Model with and without data augmentation	15
4.2.2 Model 2: 4-Layer model and adjustments to reduce overfitting	15
4.2.3 Model 3 for “adjusted” dataset with data augmentation.....	16

4.3 Creation of an Executable	16
4.3.1 Tkinter.....	17
4.3.2 Kivy	17
4.3.3 NodeJS	17
4.4.4 Python Flask.....	17
4.4 Testing custom data	17
5.4.1 Data acquisition.....	17
5.4.2 Testing Models 3a, 3b, and 3c and model 4a.....	18
4.5 Finalising app implementation and design.....	18
Chapter 5: Results.....	19
5.1. Dataset Preparation	19
5.2 Model Creation.....	21
5.2.1 Model 1: Testing 2-Layer Model with and without data augmentation	21
5.2.2 Model 2: 4-Layer model and adjustments to reduce overfitting	23
5.2.3 Model 3: for “fabrics_adjusted” dataset with data augmentation	28
5.3 Creation of an Executable	34
5.3.1 Tkinter.....	34
5.3.2 Kivy	35
5.3.3 NodeJS	35
5.3.4 Python Flask.....	37
5.4 Testing custom data	37
5.4.1 Data acquisition.....	37
5.4.2 2 Testing Models 3a, 3b, and 3c and model 4a	37
5.5 Finalising implementation and design of the Executable.....	40
Chapter 6: Conclusion and Discussion.....	41
6.1 Objectives	41
6.1.1 Change to initial objectives	41
6.1.2 Objectives met.....	41
How code reuse affected the project.....	42
Problems encountered	42
Conclusion	43
Chapter 7: Reference List	44
Chapter 8: Appendices	45
8.Appendix A: Project Definition Document	45
Project Definition Document.....	45
Proposal.....	47
Problem to be solved.....	47

Project Objectives.....	48
Project Beneficiaries.....	48
Work Plan	48
Project Risks.....	49
Risks to your project.....	49
Risks that your project poses to others.....	50
References:.....	50
Appendix: Work Plan (Gant Chart)	51
Appendix: Research Ethics Review Form.....	52
8.Appendix B: Reuse Summary.....	55
8.1 Source Code and instructions for use of notebook files and web app.....	56
8.2 Notebook files model related diagrams	60
8.3 Website pages	61

Abstract

The project aimed to create a convolutional neural network model for the classification of different fabric types with an accuracy of at least 70%, measured by the number of images correctly classified from a test, and deploy it on an easy-to-use platform for common individuals.

The goal is to enable quick and easy access, for a person without expertise in fabric identification, to a tool, which can perform the classification for them on demand.

Additionally, are insights gained documented to support the development in this area of image classification and support other individuals who have started to investigate this field.

The project was successful, producing models with over 70% accuracy for the 6 fabric types “cotton”, “denim”, “nylon”, “polyester”, “silk” and “wool”, chosen based on the number of samples available, showcasing the effects of data augmentation, adjusting the model’s capacity with dropout and L1 regularisation, adjusting the number of samples and possible target values, adjusting the amount numberers, and highlighting the importance of providing a sufficient amount of data to a model. It also produced piece software by providing the model’s fabric classification capabilities through a hosted web service, accessible through a majority of devices, such as desktop or mobile, only restricted by the access to the internet and a device’s ability to upload images.

It documents the work performed, results obtained, insights gained, problems encountered, and their impact on the decisions made to arrive at the current output.

Chapter 1: Introduction

1.1 Problem solved

In the modern-day, there are a variety of options available when it comes to selecting a piece of fabric, each type possessing unique traits such as its feel to the touch, ability to absorb and repel water, friction, and durability. Given the attributes, fabric possesses it becomes important to choose the correct fabric or combination of different layers of fabrics to make clothes to fulfil a certain need.

However it is not easy to identify fabric, based on touch on other factors, to guarantee the authenticity or to identify it if no other information is available, especially for an individual that never used a certain fabric before, and even with experience, there are no 100% certain ways to identify fabric, even the most proven method for a fabric identification “the burn test” is not a reliable way for everyone to identify a fabric given the fact that not all fabrics burn differently. (James V. 2021)

To come close to a solution for this problem, an individual would be required to learn about the ways to identify all of the different kinds of fabrics and remember each distinguishing feature to identify it, including the way it behaves when burning.

In addition to that, they may not have the option of burning a piece of fabric since the flames are a risk to their surroundings, and the fabric they want to test might be too scarce and/or expensive to cut a piece out of to perform the test, assuming they are in a workshop, clothing store or other location with flammable objects nearby.

The project aims to provide a software solution to identify fabrics based on an image taken of the fabric in question and show the result to the user, all within an easy-to-operate graphical interface that does not require knowledge of machine learning models to use.

This is achieved by using a series of images belonging to individual fabric types to train a convolutional neural network (CNN) documented to benefit further research into the topic of fabric classification and machine learning in general and provides the model’s capabilities through a web app.

1.2 Project Objectives

1.2.1 Main Objective

The project aims to create an image classification model using a convolutional neural network, with an accuracy of >70% for selected fabric types, where the figure of 70% was chosen based on the average obtained within the report of C. Kampouri et al. (2016) a computer vision project that used the same dataset.

1.2.2 Additional Objectives

1. The final classification model will be accessible through a hosted service or standalone application, that will be provided through a desktop, mobile or web application that will allow any user to load an image of their choice, which will be classified using the model and have the result returned to the user.
The application should be easy to operate by keeping the number of user-required actions to a minimum.
2. Different models and model options (e.g., adjusting layers and the amount thereof and changing the activation function) will be tested to identify the model with the best accuracy for the classifications.
3. Artificially increase the dataset size by translating images (rotate, scale, add a colour layer), if there is only a small collection of certain images in the dataset for certain fabrics, and test the effects on the data.

1.2.3 Optional Objectives

- a. Improve the dataset by gathering additional images for fabrics that lack mane/any image sets (e.g., collect images for “fur”) and retrain the model to include them
- b. Include more information about the fabric detected, such as additional checks that can be performed through touch or “the burn test”.
- c. Identify the conditions in which photos taken will yield the best results (distance, lighting, image specifications)

1.3 Beneficiaries

The project’s output will allow hobby sewers and commercial sewers alike to classify images on demand if they are unsure about the type of fabric and are unable to identify it by themselves.

The research, insights gained, and work conducted shown will also benefit other researchers who are interested in addressing this problem domain and learn from the experiences detailed throughout this document.

1.4 Work Performed

From the objectives in [Introduction \[1.2\]](#) all the optional and additional objectives have been completed, where multiple models have been created and tested with different types of changes between them and compared to identify the cause and effect of certain changes and documented with the help of graphs produced to better understand the outcomes, define a model as satisfying to the project’s main objective.

Research has been conducted (see [Literature Review \[3\]](#)) to ensure an understanding of the results obtained, options avail, able and the limitations of this project.

Care was taken in the creation of a platform to deploy the model in a user-friendly environment and ensure secure use for the user and the service host by observing other platforms and their design choices (see [Results \[1.5\]](#)).

1.5 Assumptions made

For this project it was assumed that training time will be longer, compared to the author’s previous experience with neural network models, however it was greatly underestimated, as addressed in [Conclusion and Discussion \[6\] Problems Encountered](#).

Chapter 2: Output Summary

Note: Line numbers for code are close estimates due to whitespace comments and line breaks.

2.1 The fabric datasets	
Summary	<p>“fabrics_simple” and “fabrics_adjusted” subsets of the original dataset by C.Kampouris et al (2016) and “homemade_photos”</p> <p>“fabrics_simple” (250MB) consists of 1008 images categorised under the 2 fabrics Denim (with 648 sample images) and Wool (360)</p> <p>“fabrics_adjusted” (1GB) consists of 4692 images categorised under the 6 fabrics, Cotton (2352), Denim (648), Nylon (228), Polyester (904), Silk (200), and Wool (360)</p> <p>“homemade_photos” (100MB) contains 2 subfolders for original and cropped pictures taken through phone and camera 11 times each</p>
Usage	Created in Results [5.1] , used for classification in Results [5.2] - [5.5] . “homade_photos” created and used for final tests in Results [5.4] .
Recipients	Author for continuation and other researchers for replication and reuse
Link	[Google Drive] , [Dropbox]
2.2 Model files	
Summary	<p>12 main recordings of model weights stored in h5 files and 15 model creation ipynb notebook files (~20 lines by tensorflow.org for initial model structure).</p> <p>Files were named in the format: Model-XY-(No-/Aug)-(fs/fa).(ipynb/h5)</p> <ul style="list-style-type: none">• X references the set number of a model, matched to Results [5.2.X].• Y (optional) references a change applied to a model of the same set e.g. one has a dropout layer applied and is stored where with Y=a while the model in the same set that uses L1 regularisation is named with Y=b.• NoAug (no augmentation) or Aug (augmentation) denotes if the model applies augmentation.• fs (“fabrics_simple”) or fa (“fabrics_adjusted”) denotes if the model uses the simple or adjusted dataset.• -ext which indicates that more epochs used• -lr=xe-y denote a decimal value that the learning rate is set to, other than the default rate lr=1e-3 (0.001) <p>“old_model_files” contains an old version of the current ipynb files, early training attempts, and initial approaches, the only majorly distinct files are for model 3, which followed an unstructured working methodology, blindly attempting different adjustments.</p> <p>“old_model_saves” contains the h5 for the old model 3 ipynb files</p>
Usage	Produced/Showcased in Results [5.2] , h5 files additionally used in Results [5.3] - [5.5] .
Recipients	Author for continuation, other researchers for replication and reuse
Link	[GitHub] (ipynb files), [Google Drive] (h5 files), [Dropbox]
2.3 The fabric classification app(s)	
Summary	The project produced a complete web app developed in python with files:

	<ul style="list-style-type: none"> • “image_check.js” (19 lines by the author): Checks contents of the file input field for a valid file extension and enables/disables the upload button accordingly. • “style.css” (83 lines by the author): Styles the page • “index.html” (54 lines by the author): An HTML template that gets adjusted with flask when the server sends the HTML response. • “app.py” (32 lines of code by the author + 14 lines from tensorflow.org) + 30 lines each written or adapted from existing literature by the author): runs the server, loads the model, and performs classifications tasks. <p>Two other, nonfunctionally incomplete, applications were produced: “fabric_app.py” (60 lines by the author + 14 lines of code from keras.io): A python Tkinter implementation that produces a simple GUI</p> <p>A node.js web app with files:</p> <ul style="list-style-type: none"> • “form.html” (22 lines by the author): Html for a file submission form. • “fabric_app.py” (29 lines by the author + 19 lines from tensorflow.org): Loads a model and classifies an image. • “index.js” (43 lines adapted from multiple third parties): Runs the server and loads the “fabric_app.py” script.
Usage	Created and refined in Results [5.3] and [5.5] .
Recipients	Author, other researchers, and end-users for usage in classification tasks on “fabrics” datasets,
Link	[GitHub] , [Google Drive] , [Dropbox]
2.4 Utility Files	
Summary	<p>“my_utils.py” (44 lines by the author + 60 by multiple third parties): contains several functions of code frequently reused in the model ipynb notebook model creation files.</p> <p>There exist 3 ipynb files that serve to visualize the number of samples for each fabric of the above fabric datasets: “Dataset_Analysis(‘fabrics’), “Dataset_Analysis(‘fabrics_simple’)” and “Dataset_Analysis(‘fabrics_adjusted’)”</p>
Usage	Results [5]
Recipients	Author for continuation, other researchers to execute the author’s ipynb files
Link	[GitHub] , [Google Drive] , [Dropbox]

Chapter 3: Literature Review

This project requires a good understanding of neural networks and their adjustable parameters, to be able to reason the effects, and parameter changes had on a model, to and improve one’s ability to detect when a model is performing well.

This chapter highlights the main sources of information used, in the form of written literature, video, and similar projects, the improve the author’s capabilities to perform well in this project.

Machine Learning

Machine learning belongs to the science of Artificial Intelligence, it is the concept of a system that can learn and improve itself without the help of a third party to program those improvements into it and allows a stage of improvement without being initially programmed at that stage (Byju’s Gate, 2022). They become more accurate the more data they acquire.

Deep Learning

Deep learning exists as a subset of machine learning, where multiple processing layers are used to learn the representations of data in multiple levels of abstraction, this distinction allowed it to improve the capabilities of AI in problem domains such as image and voice recognition.

It does so by creating a neural network and readjusting the internal parameters called the weights, which exist for each connection between nodes across layers, each weight, which is usually a real number in the range of 1 to -1, has an impact on the output value, by being multiplied with a node output of a node up to the output node, where nodes are readjusted each training cycle to in a process called backpropagation (Lecun Y. et al, 2015).

Convolutional Neural Networks (CNNs)

A neural network is a network of nodes similar to neurons in our brain, each node can send a signal to another node it is connected to with a certain strength called the weight.

As nodes are grouped in layers when a node receives the inputs from its connected nodes from the previous layer or the raw input if it is at the first layer it will decide which node to signal next, where the signal value is dependent on the outcome of an activation function and weight value, this process continues until the output layer is reached and repeated for many epochs employing backpropagation to reach a stage where accuracy is maximized (Code Bullet, 2018).

In addition to neural networks, there exist convolutional neural networks (CNNs) which are designed to process data that comes in the form of multiple arrays, such as images that can be processed as 2D arrays containing the RGB (red, green, and blue) values for a pixel. It is designed to identify features in an image based on locality e.g., it detects an image with a red balloon by detecting that there is a large number of adjacent pixels with a very similar red colour located in one part of an image. (Lecun Y. et al, 2015).

Similar Work

Among the vast amount of projects in the field of computer visions, there exist projects which target a similar problem domain, such image classification of the woven structure of fabric by Yasith Sanura Perara (2020), which uses a dataset, created for a project by C. Kampouri et al. (2016), which is the same dataset used for this project, both reports provided useful insight in fabric classification, CNNs, and utilities, such as callback functions to save the best performing model.

Data expansion through augmentation and Fabric identification

Any model's performance is dependent on the quality and quantity of data it receives, to learn the distinguishing features of sample types.

In the dataset for this project, problems exist in the form of having a limited amount of data for some fabric types, to address this problem it was identified that the dataset has to be adjusted and augmentation may need to be applied. For this reason, an article by Arun G. (2021) has been read, about data augmentation, its effects on data, the consideration of invalid augmentations, and the limitations and problems that may arise with certain augmentation techniques, from this article, it was identified that rotation is a valid form of augmentation as the images can be encountered under these conditions in the real world, and while flipping seems very similar, as the fabric classification focuses on very detailed features there runs the risk that some fabrics possess features of very high detail such as the direction that a string is twisted in, additionally other forms of augmentation are under consideration such as noise to mimic worse camera qualities and brightness to mimic other real-world conditions not represented in the dataset.

To identify fabrics in the real world the most proven method of identifying a fabric is the burn test, additionally, a fabric can be made out by its touch and other detail, as explained in an article by James

V. (2021), which provides information on the ways fabrics differ, from their structure, their behavior under certain conditions and the way they are commonly knitted or woven.

Creating and adjusting a suitable Model

Aside from the above literature, the basis for techniques and methods that the author used, such as creating a basic model, making training, and testing sets, and visualizing results will come from what was learned from the “Introduction to Artificial Intelligence” module that was taken at City, University of London.

For any additional research, [tensorflow.org](https://www.tensorflow.org), [tutorialspoint.com](https://www.tutorialspoint.com), keras.io, and forums such as [StackOverflow.org](https://stackoverflow.com) used for specific pieces of information in the creation of convolutional neural networks for image classification.

L1 Regularisation

At a later stage during the project additional research was conducted on L1 regularisation, due to the authors “Introduction to artificial intelligence” lectures lacking detail and elaboration on the subject, this was done to improve the author’s understanding of its operation and its impact on a model.

The most helpful article was the one produced by Kurtis Pykes (2021) which describes L1 and L2 regularisation through their use on neural networks, effects and differences, and clarified that regularisation, be it in the form of L1 or a dropout layer, is a method to reduce a models capacity to learn and memorize the training data reducing its ability for to only memorize the training data and forces the model to generalise features to perform classification tasks.

L1 Regularisation adjusts weights in a model by changing them to 0, it does so to remove weights for unnecessary features and outliers which applies an additional layer of feature selection to a model. In addition to that, L1 regularisation aims to be sparse, meaning its goal is to reduce as many weights as possible to 0.

L2 Regularisation on the other hand only pushes weights near 0 and does not directly partake in feature selection.

Chapter 4: Methods

Software development processes

To complete this project, the author followed an iterative approach, where the author researched, tested, and then analysed the results to evaluate the next steps to undertake, using this approach problems could be identified and addressed as early as possible.

This approach was chosen as the primary objective of this project is to create a satisfying model for the “fabrics_adjusted” dataset that can be expanded upon in the future with insights gained and therefore requires a record of tests performed, additionally, does this allow for quick assessments of current the results obtained compared to past results and derive logical conclusion and next steps to take.

The research, tests, and notable results have been documented and recorded using figures for the model creation process of this project.

The creation of an application puts focus on user-centred design.

Management Tools

To track progress and stay within deadlines a work plan was established in the form of a Gant chart (see [Appendix \[8.A\] Work Plan](#)) created by the author and aided in the rescheduling for delays and work to perform given the time constraint and avoided risks of missed deadlines.

To avoid the risk of data loss throughout the project files are either backed up or version-controlled through google drive and/or GitHub.

Jupyter notebook was used to display and store model figures for ease of collation of models and results.

Data Acquisition

All the original data stems from the “Fabrics” dataset obtained from I-bug by C. Kampouri et al. (2016) consisting of 7881 images taken using a photometric stereo sensor in a controlled environment.

This dataset was split into:

“Fabrics_adjusted”: A subset of the “Fabrics” dataset consisting of 4692 images of fabrics with 200 or more images, it includes Cotton (2352), Denim (648), Nylon (228), Polyester (904), Silk (200) and Wool (360). Model performance success will be measured on this dataset.

“Fabrics_simple”: A subset of the “Fabrics” dataset consisting of Denim (647) and Wool (360).


Used in [Results \[5.1\]-\[5.2\]](#).

“homemade_pictures”: A small custom dataset consisting of 22 images of cotton only (11 by camera and phone each), once in their original size and cropped to 400x400.

See [Methods \[5.4.1\]](#) and [Results \[5.4.1\]](#) for details

Model Adjustments

To refine a model and improve it several options were available such as the ones given on a slide from the author’s Introduction to algorithms course taken at city, university of London.



Neural Network training summary

<ul style="list-style-type: none"> ✓ Defined by your problem: <ul style="list-style-type: none"> ➢ Inputs ➢ Output Layer ✓ Architecture: <ul style="list-style-type: none"> ➢ Hidden Layers ➢ Nodes in each hidden layer ➢ Activation functions ✓ Hyper parameters <ul style="list-style-type: none"> ➢ Learning rate ➢ Epochs 	<ul style="list-style-type: none"> ✓ Convergence: <ul style="list-style-type: none"> ➢ Early Stopping ✓ Training: <ul style="list-style-type: none"> ➢ Loss ➢ Batch size ✓ Regularisers: <ul style="list-style-type: none"> ➢ Dropout ➢ L1/L2 regularisers
---	---

25

Figure 1 Introduction to Artificial Intelligence lecture slide from city university of London

Inputs

The image dataset, increasing its size improves a model's learning capabilities, alternatively, this may be improved through an artificial increase in data as explained in the [Literature Review \[3\]](#) however due to time constraints only rotation was evaluated for use as an augmentation option and applied.

Hidden Layers

The project quickly reviewed a 2-layer model to evaluate the effects of augmentation on the model, after which progress was continued on a 4-layer models

Nodes in each hidden layer/Filter sizes

The model uses both Nodes and Filters where Filters are unique to convolutional layers, a filter dictates the size of the image subareas in each image to evaluate such that the model learns from the features present in each subarea the more detailed the differences features the more advisable it becomes to apply a smaller filter value.

There were no adjustments made to these values due to time constraints, for the dense hidden layers several 128 was used, for the convolutional layers a filter of 32 was used.

Activation functions

The activation function used is Rectified Linear Unit (relu) for the entirety of the project.

Learning Rate

The rate at which a model learns has been adjusted on multiple models the values tested were the default value for the adam optimizer was 0.001, 0.0005, and 0.0001 shown in [Results \[5.4\]](#).

Epochs

The epoch was changed multiple times during the project to be certain of a model's performance for testing, later increased permanently to 150 until 200, and included a callback function to store the best model performance throughout the epochs and store it as an h5 similarly to how early stopping is applied.

Loss functions

The loss function used is categorical cross-entropy that has the primary use case in multiclass classification tasks, it remained unchanged throughout the project.

Batch size

The number of training samples per iteration, the batch size used was 32 as it is one of the most advised numbers when researched and remained consistent throughout the project.

Dropout and L1 Regularisation layers

Are applied to or directly after a convolutional or dense layer (excluding input and output layers), in [Results \[5.3\]-\[5.4\]](#) dropout and regularisation values have been repeatedly tested to reduce the likelihood for the model to overfit the data.

Model Architecture

The models created in this project follow the below general format:

- 0 or 1 Sequential layer which applies the augmentation to the images in the input as defined in its the corresponding variable (only augmentation applied is rotation)
- One Rescaling layer that reformats the pixel values to a range of 0 to 255 to reduce resources required to compute the original values
- One Convolutional layer that serves as the input layer followed by a Max-Pooling layer
- 0 or more Convolutional layers followed by a Max-Pooling layer
- One Flatten layer that converts the image vectors created by the convolutional layers to a layer of neurons
- 0 or more Dense layers
- One Dense layer which is the output layer

Other shared characteristics are:

A ReLu activation function, adam optimizer, a callback function to store the best training epoch in a .h5 file based on validation accuracy and loss, where loss is calculated through categorical cross-entropy.

System setup

This project used python3 with the default python libraries NumPy, os, and re for the neural network model creation, TensorFlow GPU version 2.8.0, and the Python Imaging Library (PIL) were used.

The models were created, trained, and organized using google collab, a cloud service provided by Google for which the author bought a paid subscription to have consistent access to GPU acceleration enabled storing the code and figures in ipynb files which can be opened and run with jupyter notebook alone or through jupyter notebook through anaconda.

The webapp is produced using python flask with standard libraries, os, NumPy, and sys, and additional libraries Flask, TensorFlow, PIL and UUID.

It enables a user to upload an image and receive immediate feedback on the classification results for that image which is temporarily for classification on the server-side with security taken into account, thus the addition of uuid to sanitize filenames should client-side JavaScript fail.

While not intended for actual use and merely a by-product of experimentation, a Tkinter implementation with the same python libraries as described for the python flask implementation was created, it offers the same abilities in regards to functional requirements compared the flask implementation, but it does so locally without the need for a service provider, the design is very simplistic and fails in some non-functional requirements.

While also not intended for use, for the same reason as the Tkinter implementation the Node.js implementation uses Node 14.16.1, npm 6.14.12, express 4.17.3 and, formidable 2.0.1 along with the python libraries detailed for the python flask implementation excluding flask.

4.1. Dataset Preparation

Before a model was created, the author analysed the “Fabrics” dataset to identify possible issues, such as an underrepresentation of classes such as “fur”, which has no images that represent it, which had already been discovered previously by manually browsing through the files after the dataset had been obtained.

Finally, based on the information obtained, 2 datasets were created “fabrics_adjusted” and “fabrics_simple” which were used to train the model.

4.2 Model Creation

To develop a suitable model the datasets created “Fabrics_adjusted” which contains a selection of fabrics, chosen based on [Results \[5.1\]](#), which consists of “Cotton”, “Denim”, “Nylon”, “Polyester”, “Silk”, and “Wool” and

“Fabrics_simple” which contains only samples from “denim” and “wool” were used to test different operations on the model, such as data augmentation and measures to reduce overfitting, such as regularization and dropout layers.

To have a basis for a model to start with, a model a tutorial was followed on tensorflow.org to create a 4-Layer model and achieve a better understanding in of parameters introduced with CNN, using that model, different adjustments were tested and analysed, where each adjustment with the best performing parameter value was saved to an h5 file, additionally, separate h5 files were saved for models with and without augmentation applied when corresponding tests with and without augmentation was performed.

The metric used to identify model performance is validation and training accuracy and loss, where it has to be considered, whether the performance of the model was the result of an ill distribution of sample sizes, an example of this can be seen in [Results \[5.2.2.3\] Figure 1](#). Additionally, a model should display no indication of overfitting, which is identified as the situation where training accuracy, is substantially higher than validation accuracy, or underfitting, where training performance is worse than the performance during validation.

Generally, a model’s training and validation curves should converge at some point throughout the epochs.

Models with different types of adjustment are saved to separate notebook files, as within the notebook plots are produced at runtime that displays the model’s performance.

Plots used were Accuracy and Loss plots, and confusion matrixes.

4.2.1 Model 1: Testing 2-Layer Model with and without data augmentation

To test augmentation, two 2-Layer models were created from the 4-layer base model, one with and one without augmentation applied, where the augmentation applied was rotation with a value of 0.5, the reasoning behind using a 2 layer model was that a simpler and memory capacity wise more limited model will show greater effects to data augmentation while due to the large change in the data, while requiring a lower amount of training time to produce a result, the dataset used was “fabrics_simple”.

4.2.2 Model 2: 4-Layer model and adjustments to reduce overfitting

At this stage, the base models were trained and analysed using the “fabric_simple” dataset, after which, based on the results, adjustments to reduce overfitting were introduced.

The adjustments that were tested are the addition of a dropout layer to the fully connected layer after the convolutional layers, with the range of dropout values tested ranging from 0.2 to 0.8 inspired by the classification model by Yasith Sanura Perara (2020), and the addition of regularization layers, of which the inclusion was inspired by the authors experience

in “Introduction to artificial intelligence” module at City, university of London for which values in the range of 0.05 to 0.00005 were tested and results observed.

The expectation from the addition of these layers was an improvement in the model’s performance in terms of overfitting to the data as observed with the basic model in [Results \[5.2.2\]](#) in addition to that was the author aware that, as regularization decreases a model’s capacity, that is very likely to cause a slight decline in overall accuracy which is partially the intention of regularization as the model is not supposed to be entirely accurate on the training set as that shows overfitting.

4.2.3 Model 3 for “adjusted” dataset with data augmentation

In this phase the previously tested and analysed augmentation options, which are regularization, dropout layers and data augmentation through rotation, were trained, with the difference that instead of the “fabrics_simple” dataset, the “fabrics_adjusted” dataset was used, the logical process was to first train a smaller dataset to obtain faster results on changes and progress faster through the testing and documentation of different augmentation options.

After these tests the knowledge gained on the adjustments applied and their effects could be used to make a better informed decision on which adjustments to apply on the model training with the adjusted dataset and reducing time spent idling to wait for training to conclude, as the time due to performing model training on a Google Colab subscription is bottlenecked to train only one model at a time, while, Google Colab could not be used for tasks other than file management and text editing.

The aim of this phase was to create a model with at least 70% accuracy as per the requirements, and possibly improve the model to perform as far beyond that requirement as possible before proceeding to the next stage of this project.

4.3 Creation of an Executable

The plan for the general operation of the application that was created, is to allow a user to easily use one of the models created in [Results \[5.2.3\]](#) through a simple to use single-window application with a graphical user interface that will display a button to the user which will allow them to upload an image of their choice, after the image has been submitted to the application, it performs the classification of the image using the saved model and returns back to the user the result of the classification performed in a human-understandable format such as a text saying “This image is most likely ...” possibly showing the likelihoods or confidence values for the other fabrics that the model is able to identify.

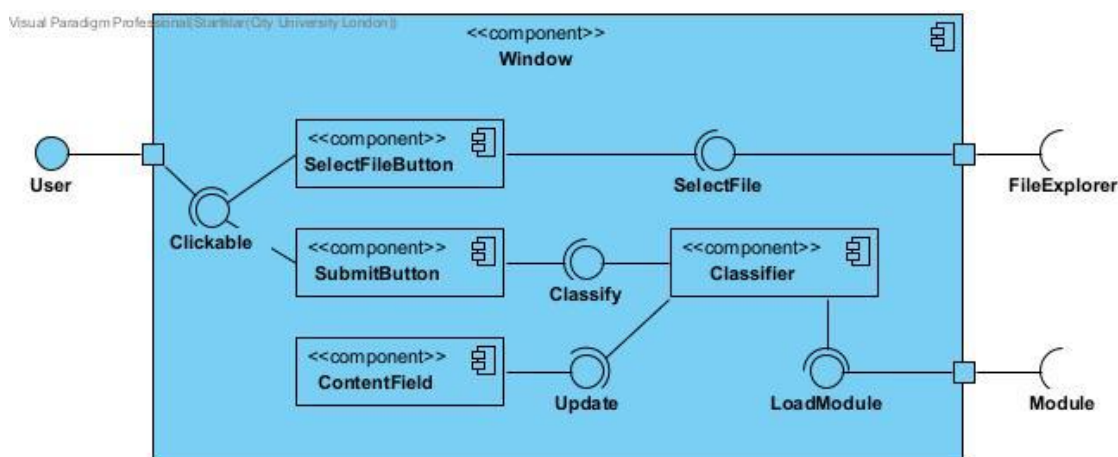


Figure 1 component diagram showcase the application's expected operation

This phase served to attempt to produce a functional operational basis for the proposed software solution.

4.3.1 Tkinter

As it is a simple GUI library to use with python, the language with which the author has the most experience with, a Tkinter implementation was made as a possible safety net for the author as the author aimed to produce a web or mobile software solution to which the author lacks experience and therefore cannot quantify an expected time investment for the implementation.

4.3.2 Kivy

As the creation of a mobile solution seemed very appealing as it made the most sense due to the increasing quality in mobile phone cameras and most custom images that would be attempted to be applied to the application probably originating from phone it was the most logical to create a mobile application, for this purpose the author searched that would allow producing an android application to be made from python code which would allow reuse of code already produced for the tkinter implementation.

4.3.3 NodeJS

After the failed attempt in the deployment of the a kivy application ([Results \[5.3.2\]](#)), it was instead decided to create a web application, as a web app offers access across multiple platforms, using NodeJS which also allows calling a python script and retrieving its output, making it possible to reuse parts of the tkinter implementation.

4.4.4 Python Flask

After conversing about the project and problems encountered with consultant and colleagues, the simple advice was taken, to produce the webapp with a python server, for which the Flask library was found and the author was able to quickly implement by adapting the node.js code.

This implementation solved all problems encountered in the node.js app which were.

- The image classification step took too long to complete, as node.js created a new child process to run the python script which involved importing the TensorFlow library, which was to time costly, before return a response to the user (~3-5 seconds).
- Instead of sending the image, performing the classification and return a result to the user, the user had to press an additional button after the image was submitted to perform the classification, which is different from how it was planned in [Methods \[4.3\] figure 1](#).
- Lack of server-side file validation/sanitation, a necessary concern when creating a webpage that is intended to host a service online to unknown parties.

4.4 Testing custom data

As the project neared its end and only a few of the optional objectives was met and the work performed on the project specifically in the model creation phases being cut, which is addressed in [Conclusions and Discussions \[6\]](#), it was decided to produce a small sample of images taken at the best and most consistent quality possible given the equipment at hand and test the performance of models produced.

5.4.1 Data acquisition

A simple set up was used to ease the process of producing images with similar quality as described in the image below.

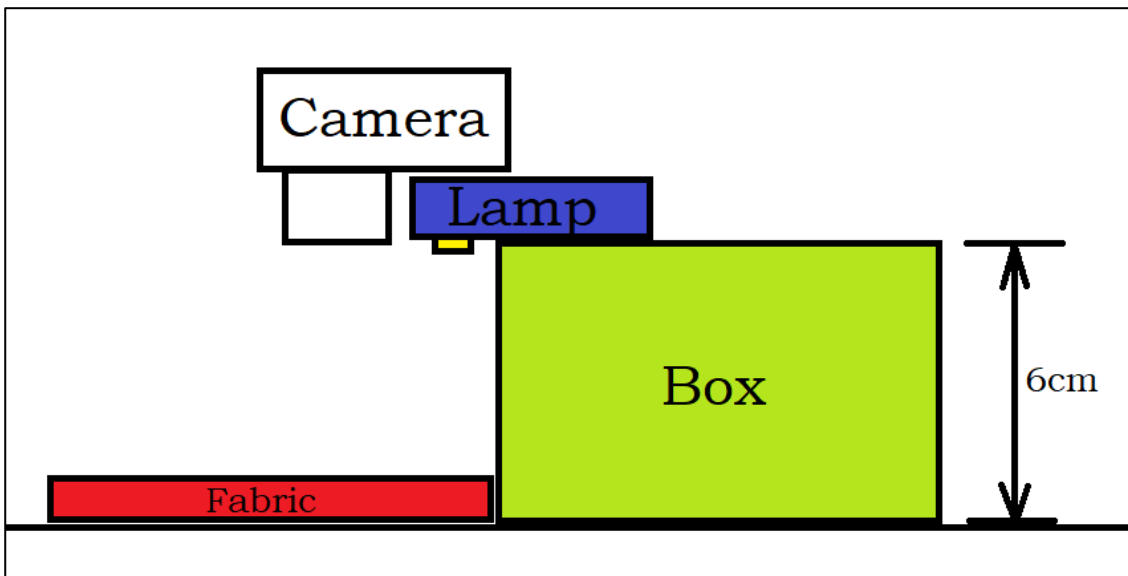


Figure 2 Camera setup for Phone and Camera

The above image shows how the camera was rested on top of a LED lamp and a box with the lens opening, reaching to the box edge which is 6cm high and the fabric was placed under the lamp and camera. This allowed images to be taken from a set distance in a lit room using a repeatable setup.

5.4.2 Testing Models 3a, 3b, and 3c and model 4a

As only a small dataset was produced and consists singularly of cotton fabrics it was decided that producing a plot such as a confusion matrix would be redundant.

In its place a table of classification results was produced for each model showing the results for the image taken with both a Sony NEX-F3 camera and Unihertz titan phone camera.

This was decided, would produce a simple overview of the results given the dataset size.

The models tested where the Model 3a 3b and 3c sets form Results [5.2.3] and Model 4a, a 6-layer model, which is an artifact from the earlier stages project, it was arrived at without a deterministic process but by simply training the model, viewing the overall accuracy and proceeding with the next adjustment that could possibly improve the result, nonetheless it was decided to be included as it achieved a ~80 accuracy in just 50 epochs.

4.5 Finalising app implementation and design

As the web app produced in Results [5.3] was functionally intact only additions for user friendliness and its completion as a full web application where conducted.

Which where to style the page and display more information back to the user such as:

- An introduction paragraph explaining the usage of the page
- Displaying the predictive power for the other fabric types.
- Displaying more information about the fabric identified to the user (as outlined in the optional objectives)

The use of ajax, to only update the page content that requires to be changed, instead of resending the entire page to populate the results was considered but lacked impact given small size of the web page.

Chapter 5: Results

5.1. Dataset Preparation

To analyse the “Fabrics” dataset and identify possible issues such as underrepresented classes and unwanted data a table was generated listing all the different categories and the number of images associated with them in an easy to interpret format.

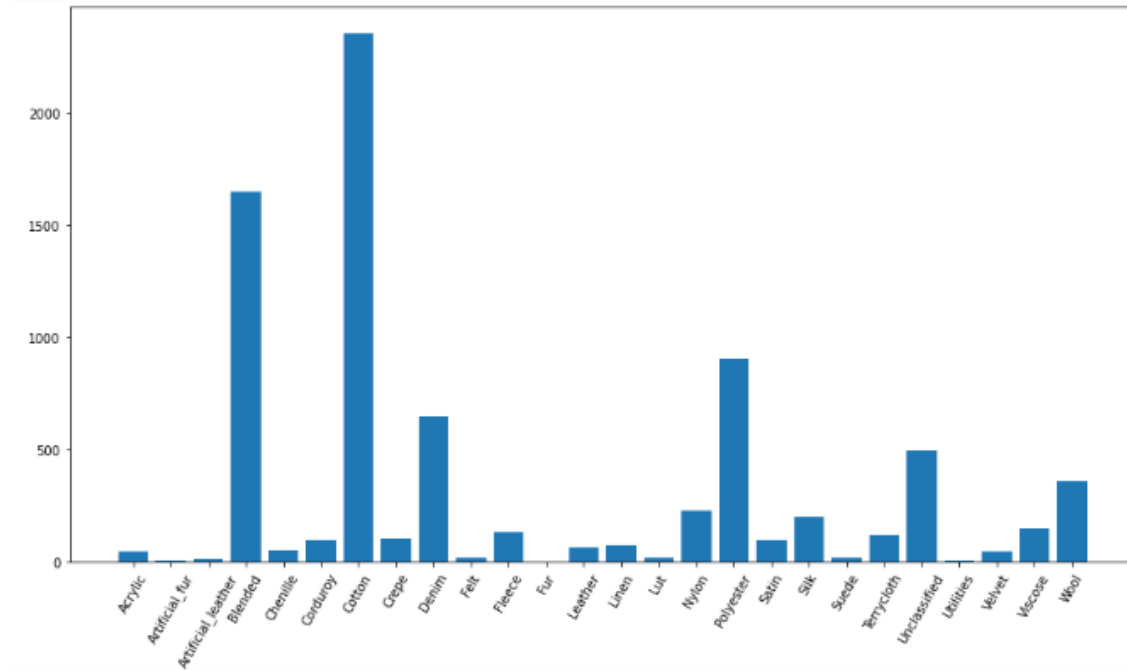


Figure 3 Spread of Original Dataset "Fabrics"

As expected there are a multitude of issues present within the dataset,

- The spread of data is very uneven, ranging from nearly ~2500 images for “cotton” to 0 for “fur”
- There is unusable data within the dataset (Blended, Unclassified, Utilities), as the model is intended to operate on non-blended fabrics.
- Many categories are barely represented, and “fur” isn’t represented by any images at all

With this information, the “Fabrics_adjusted” dataset was constructed, this is the primary dataset the model that the main model will be trained on for this project, unless additional images are added to the model during the later stages of the project, additionally, the “Fabrics_simple” dataset, that will serve as a template to observe model’s behaviour to different operations such as augmentation and measures against overfitting, has been constructed.

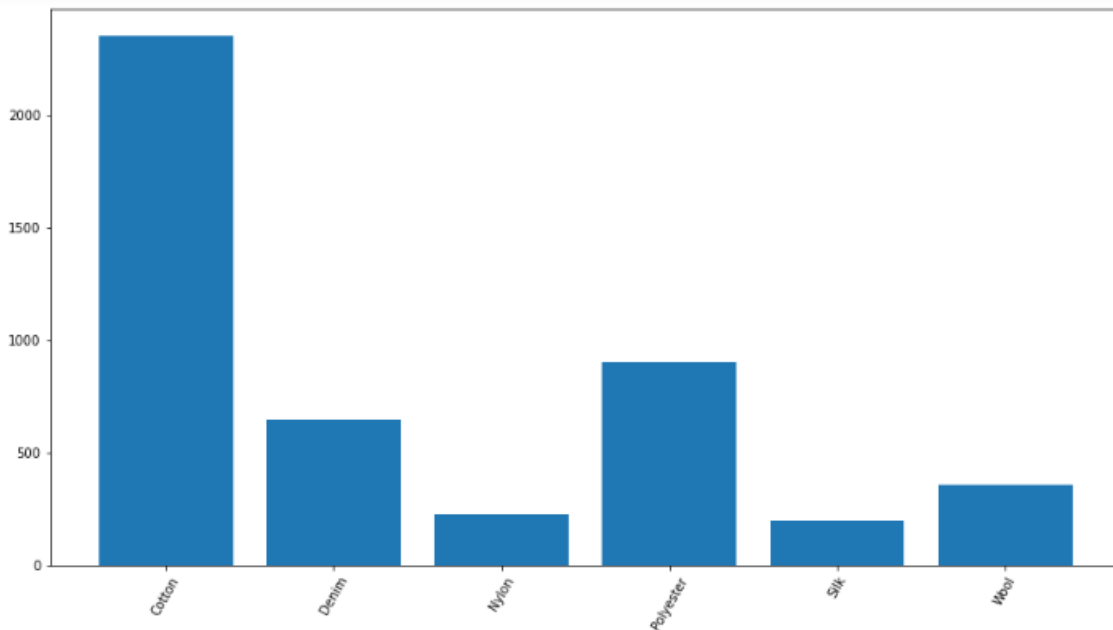


Figure 4 Spread of adjusted dataset "Fabrics_adjusted"

The "Fabrics_adjusted" dataset improves on the "Fabrics" dataset by

- Removing unusable data in the dataset which are the entirety of the categories "Blended", "Unclassified" and "Utilities"
- Removing underrepresented categories, which the author decided to be any category with under 200 images which can be observed as the fabrics in figure 1 which are below silk which has exactly 200 images.

As can be seen, the overall spread is slightly improved by removing extreme values near 0, however, "cotton" is still an outlier with over 2300 images, for this the decision was made to refrain from reducing the number of samples for "cotton" as having more images in one category still remains beneficial for the model as a whole without a downside as the model will still learn to generalize and what features are more important in the identification between fabrics even if some categories provide more data relative to other categories as it is simply more data for the model to use and overall would likely have a negative impact on the model's performance if "cotton" was removed.

With the previous paragraph in mind, the reasoning behind the decision to ignore categories with less representation is rather arbitrary in terms of requiring at least 200 samples, as the author wanted to include as many categories as possible in the model without including categories that lack representation to the point at which they are to inaccurately predicted themselves, making that specific prediction unreliable, in addition to driving down the accuracy of the model as a whole as an added side effect, and while data augmentation is to be implemented and data augmentation is a sensible solution to counter a lack of data it is not a perfect solution as the same image is reused, excessive augmentation can increase the likelihood to overfitting, for that reason the author choose to put a well-defined limitation in place for the data in-and exclude, as it has to be kept in mind that for the purposes of training the model the data will further be split into training and validation sets.

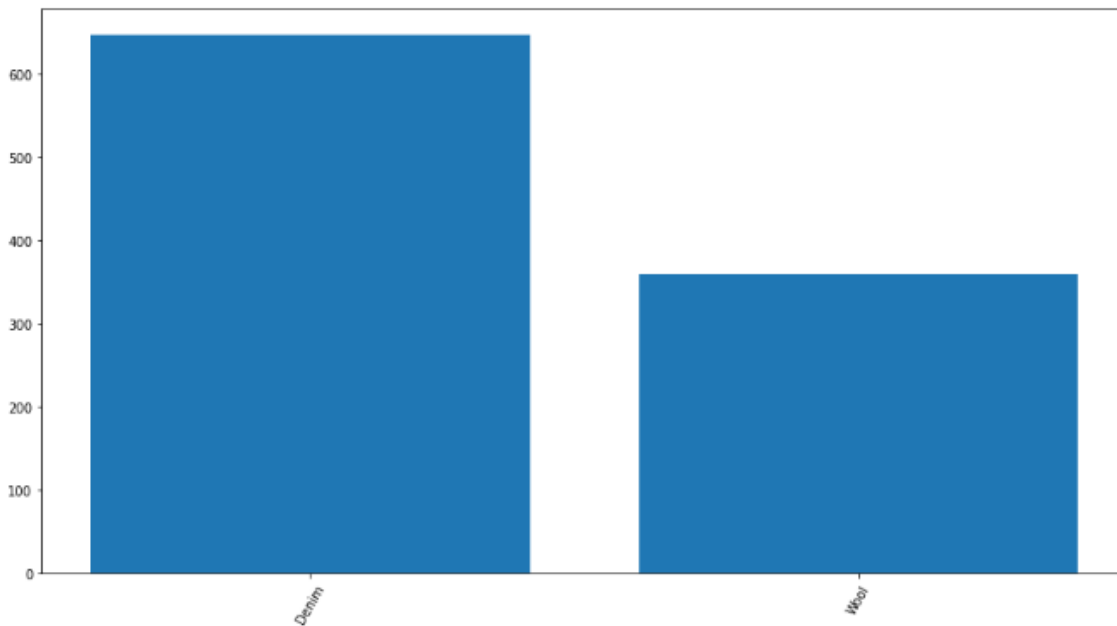


Figure 5 Spread of simplified dataset "Fabrics_simple"

The "Fabrics_simple" dataset consists of the "Denim" and "Wool" categories of the "Fabrics" and "Fabrics_adjusted" dataset, these were selected as they have a relatively large quantity of images for the model to learn from and respond to in terms of changes in the image data through augmentation and other adjustments to the model itself such as the addition of dropout and regularization layers.

5.2 Model Creation

5.2.1 Model 1: Testing 2-Layer Model with and without data augmentation

The base model was simplified to a 2-Layer model that consists of a convolutional input layer, a dense layer, and a dense output layer the model summary shows the model's structure with other layers such as rescaling and flatten as described in [Methodology \[Model Structure\]](#).

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 400, 400, 3)	0
rescaling (Rescaling)	(None, 400, 400, 3)	0
conv2d (Conv2D)	(None, 398, 398, 32)	896
max_pooling2d (MaxPooling2D)	(None, 199, 199, 32)	0
flatten (Flatten)	(None, 1267232)	0
dense (Dense)	(None, 128)	162205824
dense_1 (Dense)	(None, 2)	258

Figure 1 Model summary of model 1

Training this model with the "fabrics_simple" dataset yielded the following results with and without the augmentation layer that applies rotation to the input images.

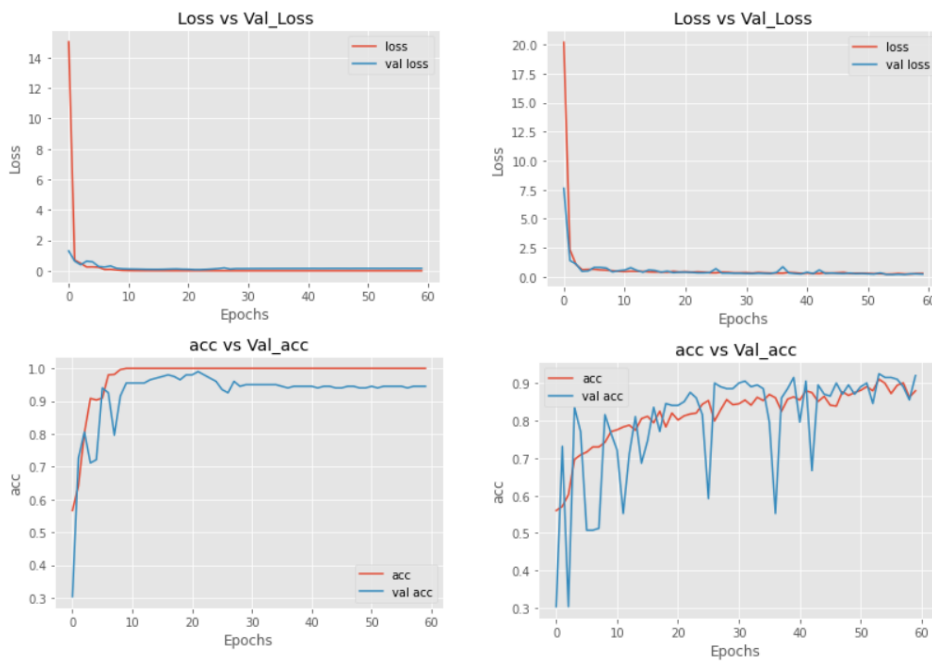


Figure 2 Model 1 without (left) and with (right) augmentation

In the above figures it is clear that augmentation has an effect on model performance, furthermore, did the augmentation performed (rotation) solve the problem of overfitting which occurred in figure 2 as the training accuracy (red) maxing out at 100% accuracy early on during training while validation accuracy (blue) concludes at 95% accuracy, although there is a peak of 99% accuracy on epoch 21, this might indicate that while training accuracy does show that the model does overfit, stopping the model early has can provide us with weights adjusted in a way that it can perform well on unseen real-world data just as in the unseen validation data.

In the model with data augmentation in figure 3, while supporting model performance and reducing overfitting seen by the validation accuracy surpassing training accuracy, validation accuracy did decrease slightly by ~5% and there are many fluctuations in the accuracy deviating by up to ~35% between epochs, therefore despite reaching an accuracy above 70% there is still reason to make further adjustments to the model in order to reduce the deviation across epochs and possibly further improve accuracy and the model's ability to generalize for real-world data.

5.2.2 Model 2: 4-Layer model and adjustments to reduce overfitting

5.2.2.1 4-Layer Model:

This is the primary model for the majority of this project, it is a 4-Layer model and has the following structure.

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 400, 400, 3)	0
rescaling (Rescaling)	(None, 400, 400, 3)	0
conv2d (Conv2D)	(None, 398, 398, 32)	896
max_pooling2d (MaxPooling2D)	(None, 199, 199, 32)	0
conv2d_1 (Conv2D)	(None, 197, 197, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 98, 98, 32)	0
conv2d_2 (Conv2D)	(None, 96, 96, 32)	9248
max_pooling2d_2 (MaxPooling2D)	(None, 48, 48, 32)	0
flatten (Flatten)	(None, 73728)	0
dense (Dense)	(None, 128)	9437312
dense_1 (Dense)	(None, 2)	258

Figure 1 Model summary of model 2

Training this model with the “fabrics_simple” dataset yielded the following results with and without the augmentation layer that applies rotation to the input images.

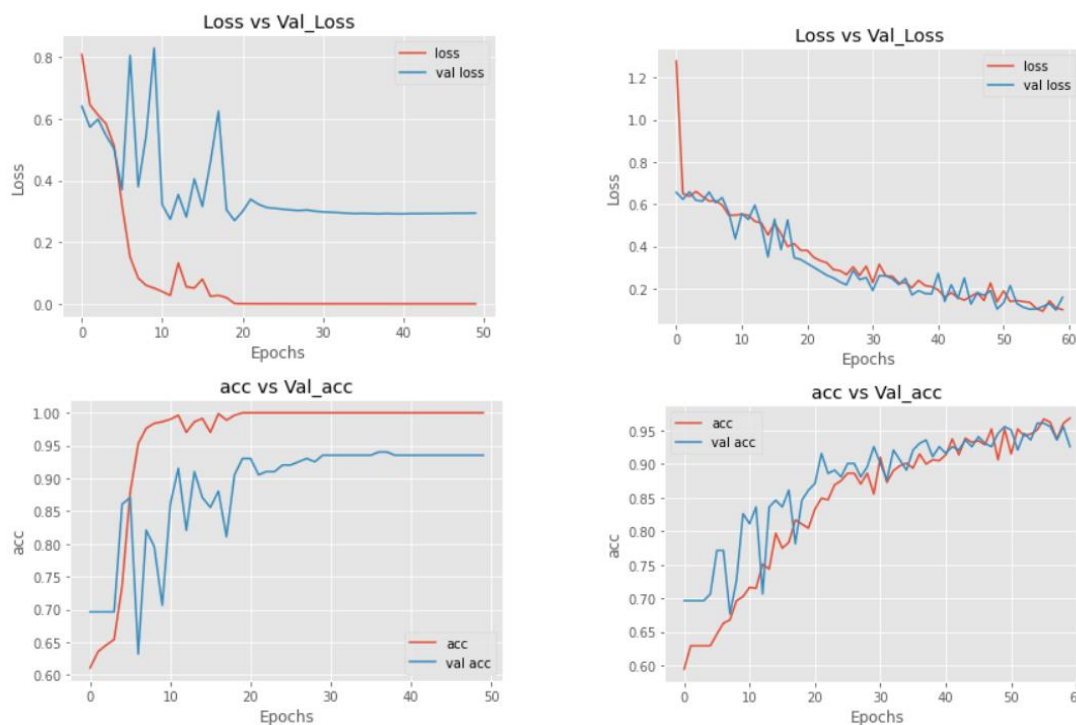


Figure 2 Model 2 without (left) and with (right) augmentation

As can be seen in the results of the model without data augmentation, a peak validation accuracy (blue) of roughly 94% is achieved however there is a high chance of overfitting that is indicated by the training accuracy (red) topping at 100% after only 20 epochs.

On the other hand, the additional layers seem to slightly improve overfitting compared to Results

[5.2.1] figure 2, where the training accuracy reaches 100% before the 10th epoch. While this model performs better it is still overfitting.

The results from the model with data augmentation, levels at around 90-95% validation (blue) and training (red) accuracy after the 50 epoch mark, as an additional note, both curves show similar behaviour throughout the epochs with the validation curve outperforming the training curve for the majority of the epochs similar to Model 1 with data augmentation from Results [5.2.1] figure 2 however the problem of large deviations in accuracy across epochs was also overcome as the largest jump in the accuracy is ~15% and after the 20 epoch mark the largest deviation shrinks down to 5% or less.

For both models, it should be noted that training and validation loss have changed significantly, compared to Results [5.2.1] figure 2, where for this model in figure 2 loss increased by 0.2, in figure 3 for the model with augmentation loss is much lower at roughly 0.1 that while worse still worse than for model 1 does show an improvement in the form of validation loss being lower than training loss.

5.2.2.2 Dropout layers

At this stage, The author added 1 or 2 dropout layers before the output layer to model 2 with dropout rates of 0.2 and 0.8.

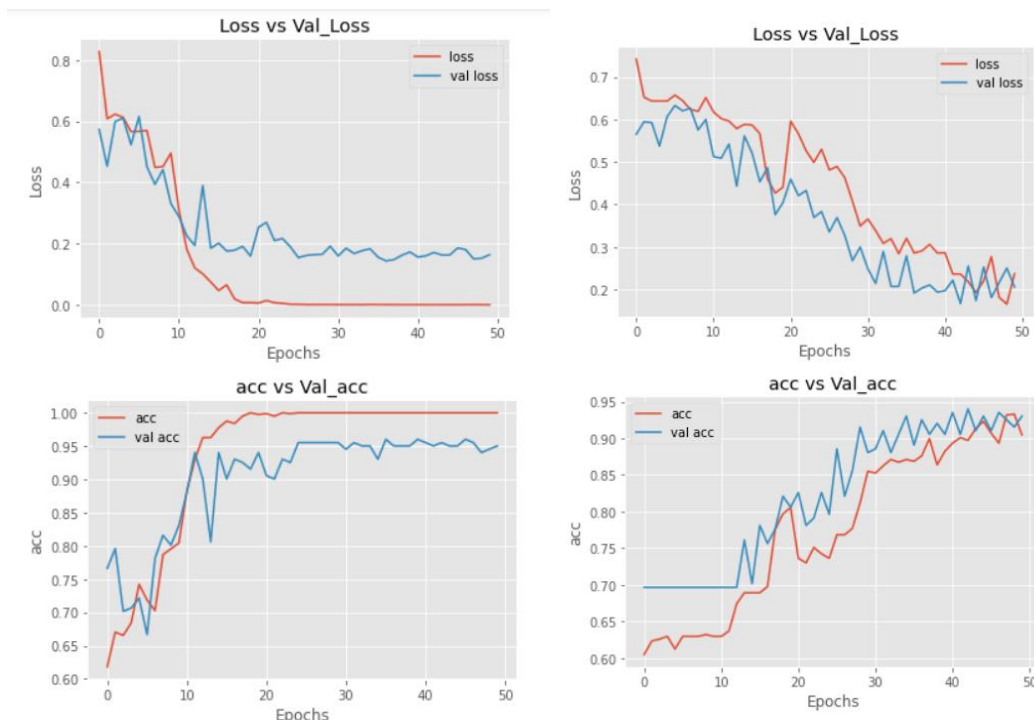


Figure 1 Model 2 without (left) and with (right) augmentation, and 1 0.2 dropout layer

As can be observed in the above figure, the 0.2 dropout layer has a small effect on the training loss and accuracy as it smoothed the transition from a curve to flatlining on the minimum and maximum values for training loss and accuracy respectively slightly.

On the other hand, validation loss (blue) has improved by 0.05 and stays near constant at ~0.18.

With augmentation the validation curves is now noticeably higher than the training curves, however, this is not due to the validation curve improving but the training curve getting worse, this indicates that the model did improve to further generalize at the cost of performing worse on the training data.

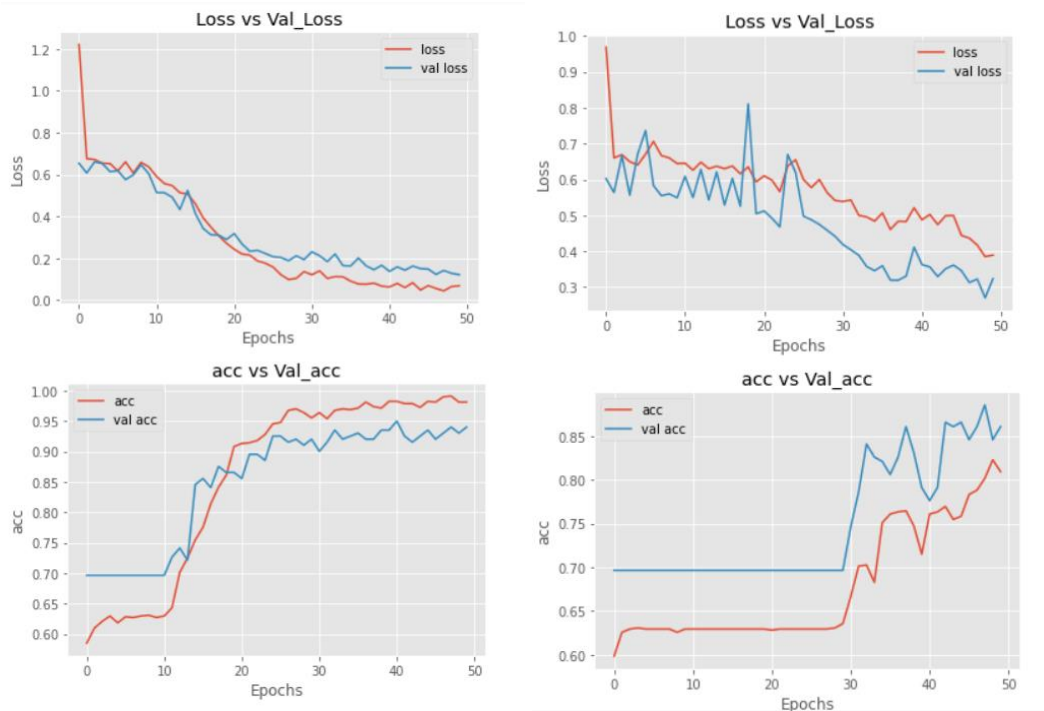


Figure 2 Model 2 without (left) and with (right) augmentation, and 1 0.8 dropout layer

The change from 1 0.2 to a 0.8 dropout layer shows similar performance values as the 0.2 dropout layer however the training curve no longer flatlines at the minimum and maximum loss and accuracy, with that training and validation curves are now closer together which could indicate that the model while performing worse on the training data, is overfitting less and has improved at generalizing which was not the case for figure 1 with a 0.2 dropout layer.

The model with augmentation in figure 2 on the other hand is shows a significant decrease in performance as but further pushing the validation curve ahead of the training curve, this indicates a first instance of underfitting where the model is unable to fit properly to the training data as the dropout layer parameter value is to extreme.

5.2.2.3 L1 regularization

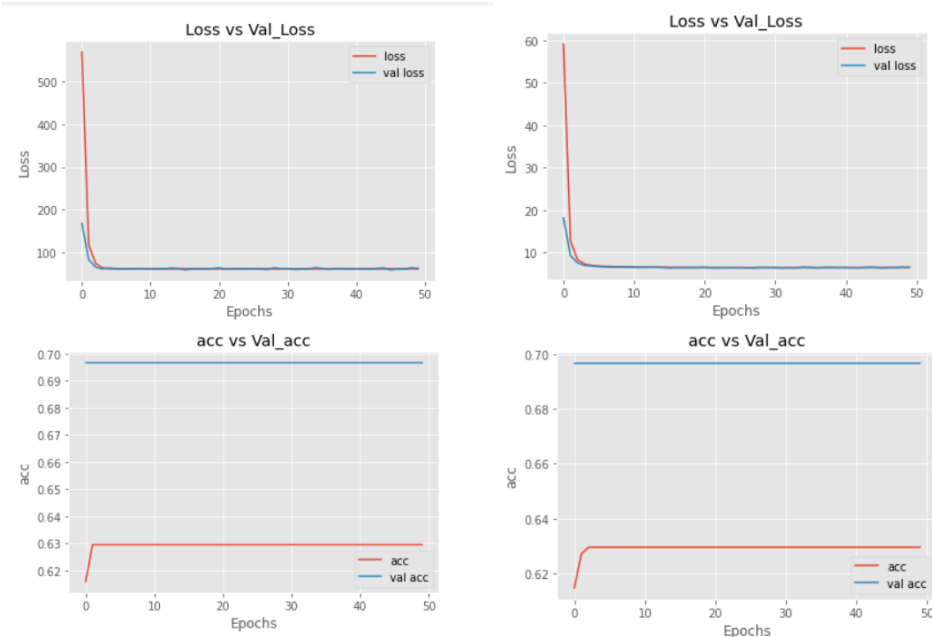


Figure 1 Model 2 without (left) and with (right) augmentation, and L1 regularization at 0.005 or higher.

As observable in the above figure, L1 regularization set to 0.005 or higher leads to an odd phenomenon as the validation curve and training curve immediately flatline and stay constant throughout the epochs regardless of the addition of artificial data.

When we view the predictions on a confusion matrix we can see that the model with these configurations does always classify an image as Denim which matches the accuracy displayed above which is ~70%, to the ratio of the number of samples for denim which is 140 over the total number of samples which is 201 which is $140/201 = \sim 70\%$, this behaviour can be explained as most likely L1 regularization being set too high, reduces the weights of actually important features to 0, which in turn, leads the model to train on a very limited number of features removing its ability to learn any distinction between Denim and Wool.

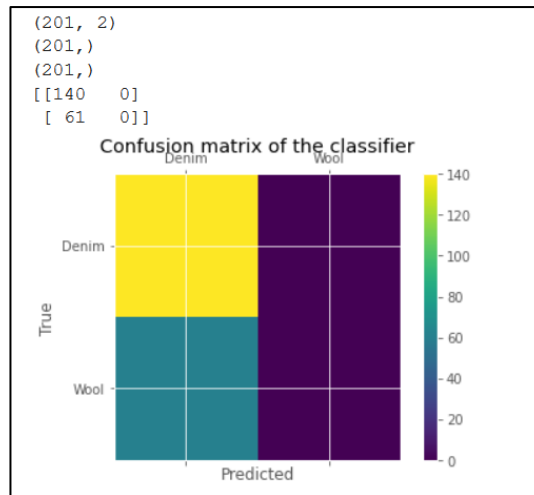
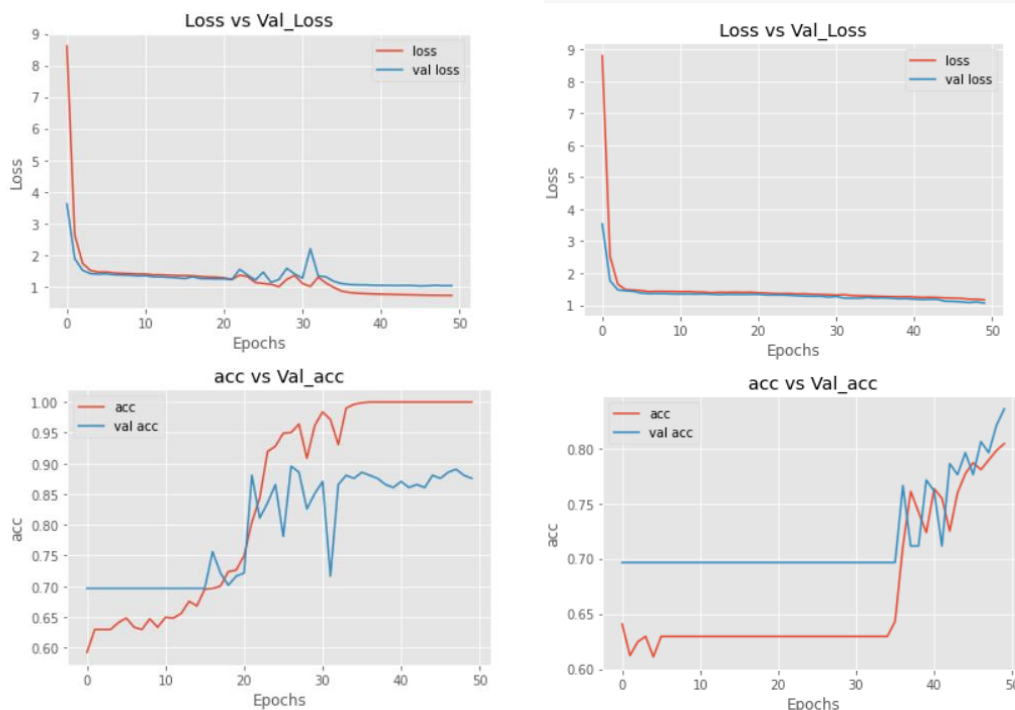


Figure 2 Confusion Matrix if predictions with models from figures 1 and 2



Figures 3 Model 2 without (left) and with (right) augmentation, and L1 regularization at 0.0005.

The addition of 0.0005 L1 Regularization shows an improvement over the other figures without augmentation for model 2 by delaying the training accuracy from flatlining and the validation accuracy becoming near-constant, until after epoch 30 instead of epoch 20, on the other hand, validation loss

increased to 1 the highest loss for model 2 that has been observed, this increase in loss corresponds to the decrease in accuracy.

As an additional note, the validation curve between epochs 0 and 14 experiences the same event as figure 1, which indicates that 0.0005 should be the borderline highest value that L1 regularization can be set at.

The model with augmentation behaves much differently, initially the same event as in figure 1 occurred but changes after epoch 35 where accuracy further increases with training and validation curves being closely aligned with each other where the validation accuracy is still slightly higher than training accuracy, throughout the epochs, loss is decreasing slowly with the validation loss performing slightly better than the training curve.

However, in the right plot of figure 3, the accuracy curves have not shown signs of diminishing or becoming constant throughout epochs before epoch 50, for that reason the author decided to observe the results for the same model with an increased number of epochs.

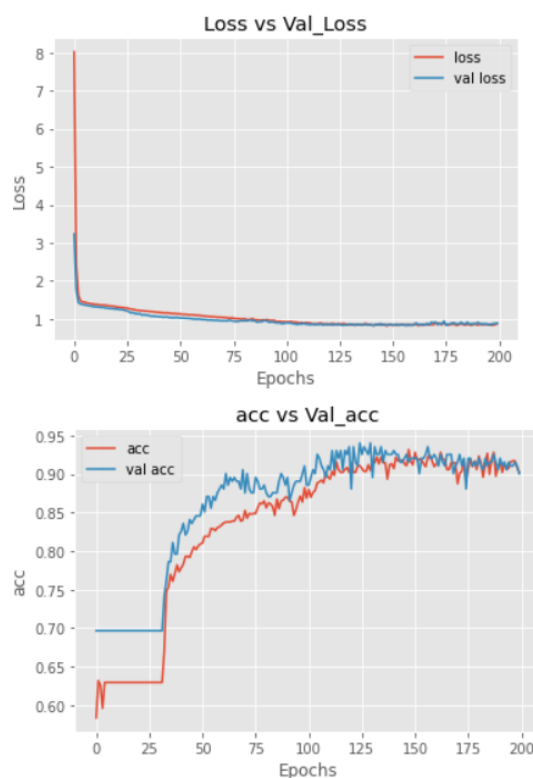


Figure 4 Model 2, and L1 regularization at 0.0005 with 200 epochs

As can be seen in the figure 4, the model continues to improve after the 50 epoch mark reaching a top validation accuracy of ~94% at epoch 125 after which accuracy and loss start to become near-constant. Overall the model in figure 4 performs well as the training and validation accuracy are high while validation is above training accuracy but training accuracy is not significantly lower than validation accuracy, additionally, the loss is still low at 1, and although more epochs have to be run to achieve higher accuracy, that is no detriment to model performance, but only requires more computational power to train in a reasonable amount of time.

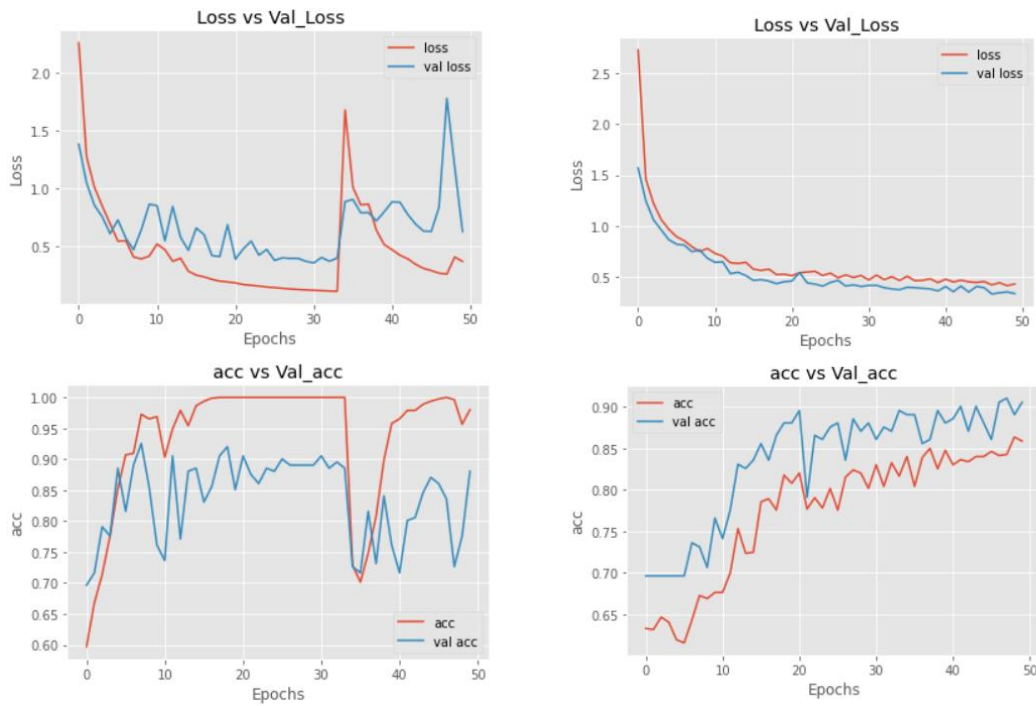


Figure 5 Model 2 without (left) and with (right) augmentation, with L1 regularization at 0.00005

In the figure above, we can observe that loss has halved in both cases compared to the plots in figure 3 where regularization is set to 0.0005, however, in the plot without augmentation (left) there is a noticeable peak and dip in the loss and accuracy curves respectively that occurs on epoch 32 with the validation curve continuing to fluctuate between 90% and 70% accuracy even when increasing the number of epochs to 100 to check if it eventually becomes more stable.

The most likely reason for this event is that the regularization reduces the weight of a certain feature to 0, with the feature that was removed is a feature that was used to a high capacity by the model in order to perform its classification task, thus removing that feature in its entirety, required the model to relearn the data without that feature.

In figure 5's model with the augmentation layer (right), the model is much more stable after the first 20 epochs, the model also performs quite well and while it would seem that the model should still improve after 50 epochs, the additional investigation conducted by increasing the number of epochs from 50 to 100 has shown no significant change apart from a very slight rise in accuracy before becoming relatively constant, averaging at 90% validation accuracy, with training accuracy roughly 3% lower and staying near constant with small rises and falls in a 3% percent range.

For both figures, a notable achievement is that validation loss in the model with augmentation in figure 5 does reach below 0.5 and in addition to being ahead of the training curves, without the training curves being significantly worse behind which lessens the concerns about underfitting.

5.2.3 Model 3: for "fabrics_adjusted" dataset with data augmentation

With the knowledge acquired from the methods applied in Model 2 which are dropout and L1 regularization, Model 3 is constructed which combines the 2, as a start, L1 regularization was to 0.00005 and one dropout layer before the output layer, with a value of 0.8 and then trained the model for 50 epochs for both datasets, "fabrics_simple" and "fabrics_adjusted".

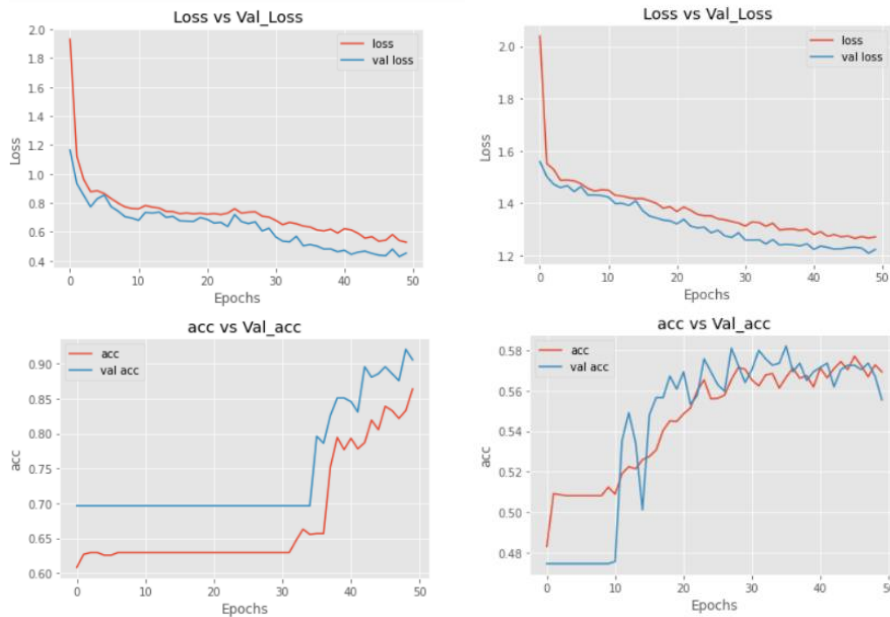


Figure 1 Model 3a results ($L1 = 0.00005$ and $dropout = 0.8$) with “fabrics_simple” (left) and “fabrics_adjusted” dataset (right)

As can be seen in the above figure the model when trained using “fabrics_simple” performs very similar Results [5.2.2.3] figure 3, however, when trained with “fabrics_adjusted”, performance decreases drastically in terms of accuracy and loss which was expected as the problem becomes much more complex the more fabric is to be identified.

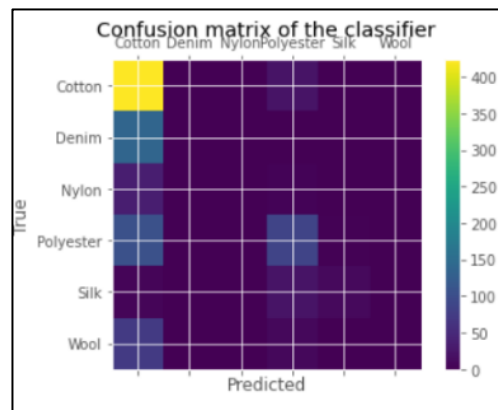


Figure 2 Confusion Matrix for Model 3a using “fabrics_adjusted”

To better understand the model’s performance, a confusion matrix was generated.

As can be seen from the above matrix, the model is able to identify some fabrics primarily Cotton and polyester, this indicates that the model has not learned the set of features or their importance for the classification, a possible reason for that occurrence could be that too much information is lost when applying dropout and L1 regularization at the values specified above.

5.2.3.1 Model 3b (no regularization and dropout applied)

In model 3b the removal of all regularization and dropout, which makes this model identical to model 2, and only keeping the augmentation layer provided the following result.

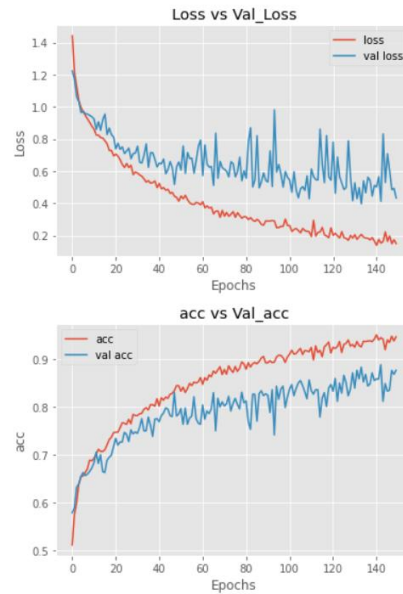


Figure 1 Model 3b (0 dropout and 0 L1 regularization)

In figure 3 we see that the removal of dropout and L1 regularization caused an improvement in loss and accuracy, reaching 80% validation accuracy and 0.7 validation loss at epoch 50 at which point it is not entirely clear if the model is overfitting by observing the accuracy, however, if we observe the loss we see the validation loss is growing stagnant much earlier than training loss which is again a sign of overfitting.

To further understand and minimize the possibility to be misled by the loss and accuracy values about the model's overall performance, the confusion matrix was analysed.

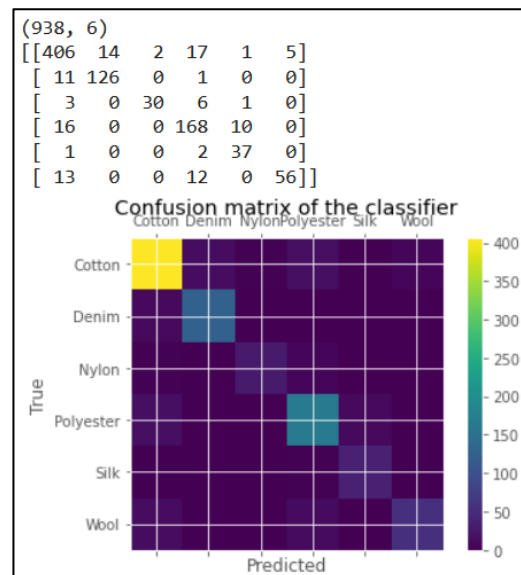


Figure 2 Confusion matrix for model 3b (0 dropout and 0 L1 regularization) after full 150 epochs

In the confusion matrix, we can see that the model is making many correct predictions, however, to fully understand the accuracy of the model, additional values were calculated for each fabric, namely the rate at which a given fabric type is correctly classified as the given fabric, the true positive (TP) rate, and the rate at which a fabric being predicted is a misclassification of another fabric, the false positive (FP) rate.

Calculated as:

TP = number of correct predictions for a group / total number of samples for that group

FP = 1-(number of correct predictions for a group / total number of predictions for that group)

Cotton: TP = 91%, FP = 10%

Denim: TP = 91%, FP = 10%

Nylon: TP = 75%, FP = 6%

Polyester: TP = 87%, FP = 18%

Silk: TP = 93%, FP = 26%

Wool: TP = 69%, FP = 8%

The data shows that the model is not disproportionately making more accurate predictions for one fabric over the rest or otherwise being predicted more inaccurately than the rest, apart from wool, which achieves a TP rate of 69%, which makes it the least reliable fabric to identify, on the other hand, if we look at the TN rate we see that a prediction of wool is very reliable to be wool as opposed to silk that is more commonly a misclassification of polyester or polyester itself which has an increased chance to be a misclassification of an image of cotton or wool.

The results of this model were very unexpected, this model shows high accuracy and low loss and does not have a skew to predict only certain fabric groups reliably. However, there is a concern that this model is in fact overfitting to the training data, in particular the imagery and controlled environment in which the pictures from the original dataset were taken.

Therefore to be certain of the performance new images would be required to be taken as an input to this model's performance in the future.

5.2.3.2 Model 3a (0.00005 L1 regularization and one 0.5 dropout layer)

After repeated adjustments to the model shown in [Results \[5.2.3\] Figure 1](#), the final configuration for the regularization and dropout layers is 0.00005 for the L1 regularization layer and 0.5 dropout on the last layer before the output layer.

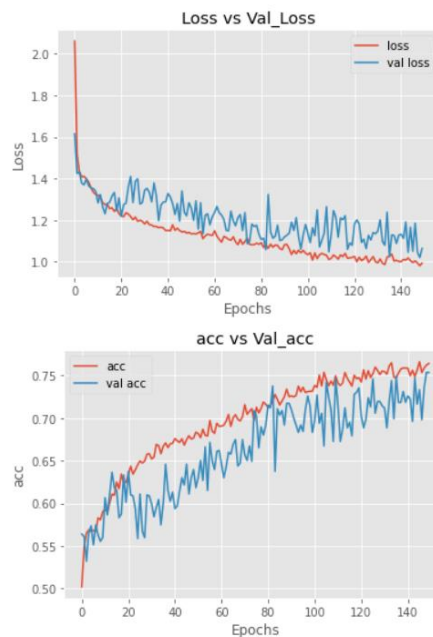


Figure 1 Model 3a (0.00005 L1 regularization and one 0.5 dropout layer)

In this model, which is the final adjustment for the model shown at the start of [Results \[5.2.3\]](#) we can see an improvement in accuracy and loss although in the first 50 epochs the right graph of [Results \[5.2.3\] figure 1](#) is very similar, with the exception of validation loss showing larger changes from epoch

to epoch and the accuracy being higher even during the start of the training. Another difference is that the validation curve does not outperform the training curves but instead stays just behind the training curves and throughout the epochs meets the training curves at multiple points during the training (e.g. in figure 1 above, the accuracy curves meet at epochs 20, 73, 80, etc.), this, additionally to the observations made in comparison with [Results \[5.2.3\] figure 1](#) is also an improvement in terms of overfitting compared to [Results \[5.2.3.1\] figure 1](#).

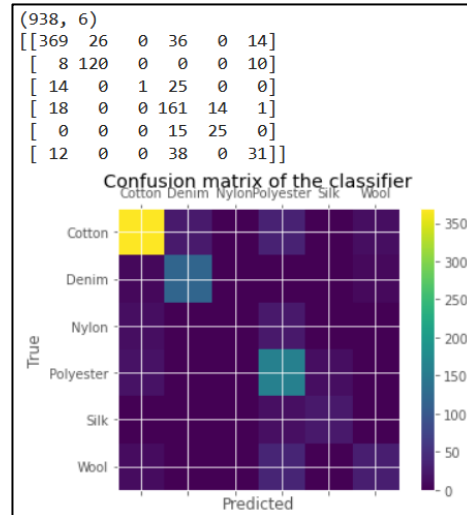


Figure 2 Confusion matrix for model 3a (0.00005 L1 regularization and 0.5 dropout) after full 150 epochs

However, if the classification of the validation data, is viewed through a confusion matrix, It becomes clear that this model is not appropriate for the classification of the selected fabric groups, alone the fact that Nylon is only correctly classified once out of a total of 50 images is enough reason to determine that this model will not be used for the application that this project aims to create.

5.2.3.2 Model 3c adjusting the learning rate

In model 3c the learning rate was changed from the default rate of 0.001 to 0.0005 and 0.0001, the dropout layer set at 0.5, and L1 regularization set to 0.00005

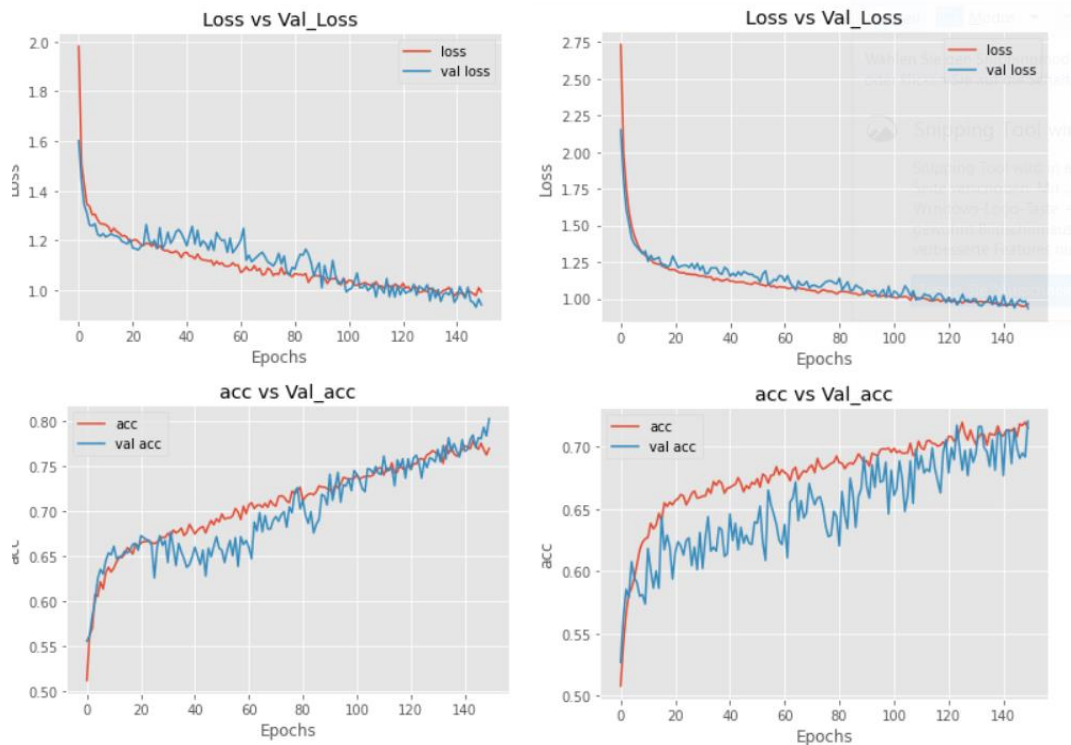


Figure 1 Model 3c with learning rate = 0.0005 (left) and learning rate = 0.0001

In the left plot of figure 1, various improvements can be observed compared to the results of other models, such as for model 2b in Results [5.2.3.1] figure 1, where this model improves on it through the removal of potential overfit as the alignment of validation and training curves for loss and accuracy has shifted closer together and are.

On the other hand, this model does show a decrease in the training accuracy, compared to the model in Results [5.2.3.2] figure 1, validation accuracy and loss on the other hand are nearly matching during the end of the training, therefore in terms of performance measured at the accuracy and loss of validation and training sets, this model performs better.

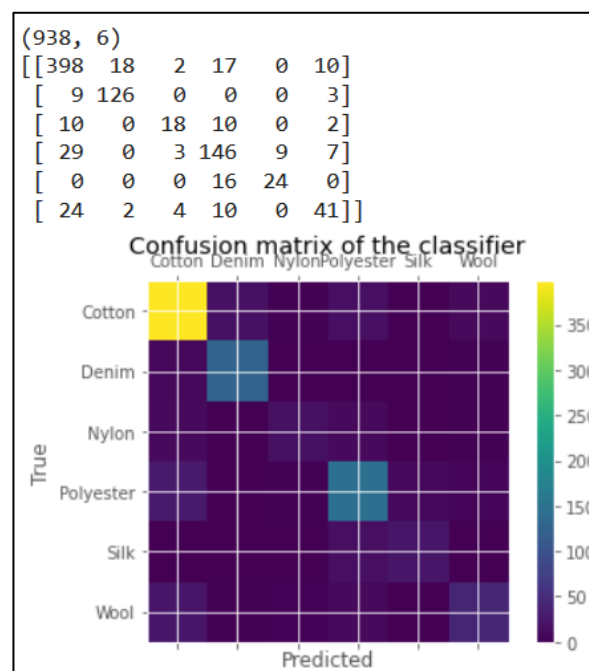


Figure 2 confusion matrix model 3c learning rate = 0.0005

True positive (TP) and false positive (FP) rates:

Cotton: TP = 89%, FP = 15%
Denim: TP = 91%, FP = 14%
Nylon: TP = 45%, FP = 33%
Polyester: TP = 75%, FP = 27%
Silk: TP = 60%, FP = 27%
Wool: TP = 51%, FP = 35%

From these values, true positive and false-positive rates, such as for denim and wool, are noticeably higher or lower compared to each other, correlating with the number of samples used for the training to identify that category, this shows that the previous decision to not remove samples from the cotton fabric samples, outlined in relation to [Results \[5.1\] figure 2](#), has less of an impact than anticipated, given by the fact that denim is the second most represented fabric and has near the exact same true and false positives rates such as cotton, which show that the access to more data, be it for one category alone, improves the model's accuracy to classify other less represented fabrics as a whole.

5.3 Creation of an Executable

5.3.1 Tkinter

The Tkinter application developed followed all the initial planning which includes the component diagram from [Methods \[4.3\]](#).

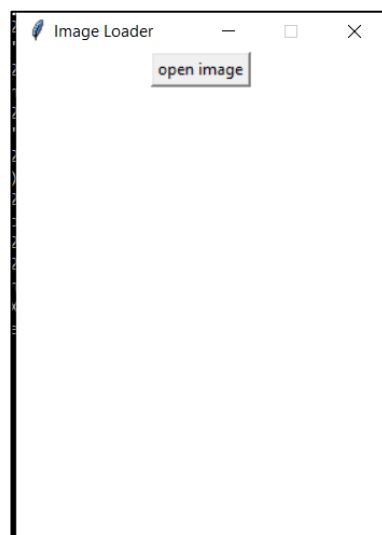


Figure 1 Tkinter application start-up page

When executed the user is presented with a single application window displaying only the option to upload an image.

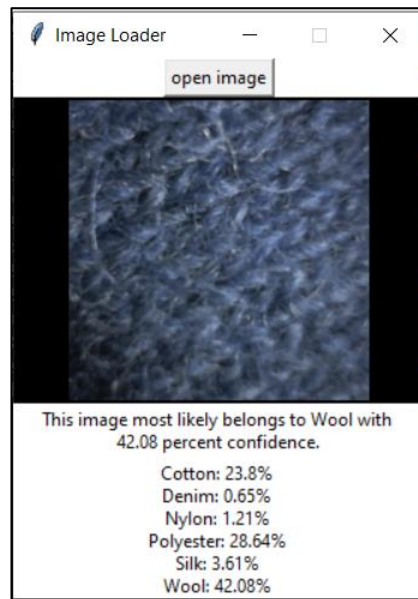


Figure 2 Tkinter page after an image was selected through the file explorer selected through "open image" button

Upon selecting an image through the operating system's file manager, the application will pass the image through the trained model. After the image is classified, the application will return to the user what kind of fabric was identified along with the confidence value in %.

5.3.2 Kivy

After more time was spent than planned on the attempted implementation of an apk using Kivy, the idea was discarded.

5.3.3 NodeJS

Using node.js a simple website was produced, which could be hosted through the author's machine and accessed locally. From the node server, the python script and model would be loaded when requested. This enabled access to the model through the internet and perform classifications on demand independent of whether it is accessed through any brand of mobile device or computer as long as it has the ability to browse the web and upload a file.



Figure 1 Node.js page file selector button and submit button(disabled)
 ("Durchsuchen" -> "Open File", "Daten absenden" -> "Submit", "Keine Datei ausgewählt" -> "No file selected")

As in the Tkinter application, the web page first displays the option to select a file.

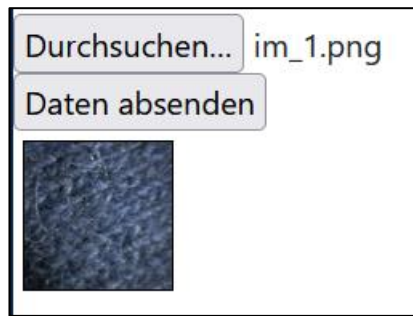


Figure 2 Node.js page with a valid image selected and submit button(enabled)
("Durchsuchen"->"Open File", "Daten absenden"->"Submit")

Upon selecting a file the image is displayed for preview and the button that allows the user to send the image for classification to the server is enabled.

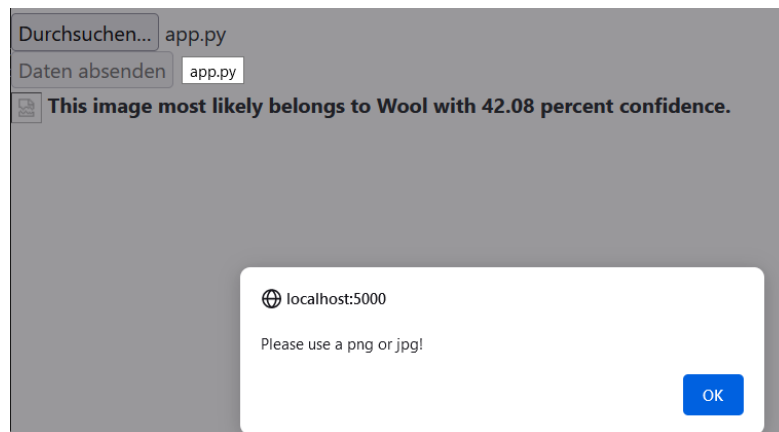


Figure 3 Node.js page alert of file with invalid extension selected

Pre-emptively a script was also added to prevent the user from selecting an image that is not of png or jpg format.

Furthermore should this service be deployed in a real environment additional server-side validation will be required to be put in place to prevent abuse by certain users to send malicious packages to the server.



Figure 4 Node.js website in between page to confirm submission and perform image classification

Upon completion of the form and its submission, a post request is sent to the server with the image, of which the image is then saved on the server side for later classification after "Evaluate on the file uploaded confirmation page is clicked

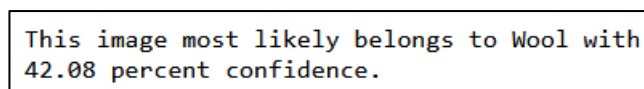


Figure 5 Node.js page with classification result displayed

After the user presses "Evaluate", the user will be redirected to a new screen once the server runs the python script which performs the classification of the image, as the script runs on the server so does

the image also have to be located on the same server which is the reason for sending the image through the browser in the previous step, meanwhile the “Evaluate” link requests a page that calls the python script passing the image name as an input using a URL query, meaning the image name is passed into the link to the page as a parameter.

5.3.4 Python Flask

Two tutorials on the deployment of TensorFlow models provided an introduction to the use of flask with python server to create a website one by Frank A. (2022) with a form to send data to a machine learning model and return the output back to the user with the help of pickle to store the model as a variable, the other tutorial was by Faizan A. (2021) which showed how to accomplish the same objective, however it included information for image classification and also showcased the use of the “secure_filename” import which sanitises the submitted file on the server side which was later replaced by changing the file to a universally unique identifier (uuid).

The combination of the 2 tutorials in combination with the authors code from the node.js server allowed all problems identified to be solved.

The web app operates identical to the node.js website, with the only distinction being, no longer requiring the confirmation page, shown in [Results \[5.3.3\] figure 4](#), and applying minor styling to the results page to results page which is simple the initial window but with a variable text field updated to display the classification result, allowing the user to directly classify another image instead of having to navigate to the start page and in addition, is the user shown the result of the classification in a matter of fractions of a second instead of several seconds as in the node.js implementation.

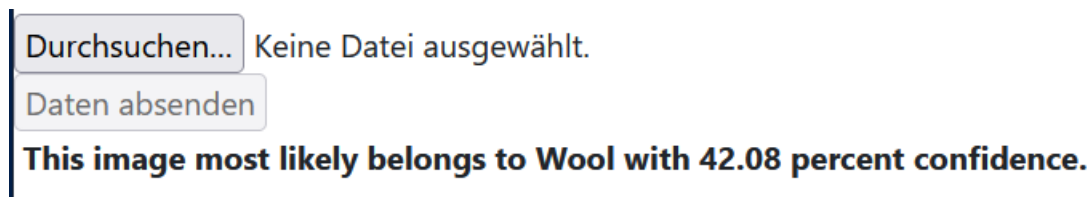


Figure 1 app.py result page with “open file” button and “Submit” button
("Durchsuchen"->"Open File", "Daten absenden"->"Submit", "Keine Datei ausgewählt"-> "No file selected")

The migration to python server successfully solved all the issues identified with the node.js server.

5.4 Testing custom data

5.4.1 Data acquisition

22 Images were taken of various cotton fabrics using a

Camera (Sony NEX-F3): Resolution 350 dpi, bit depth 24, exposure 1/60, F/22, ISO 800, flash on

Phone (Uniherz Titan): Resolution 72dpi, bit depth 24, exposure 1/100, F/1.7, ISO 100, flash on

To produce the “homemade photos” folder containing cropped 400x400 pixel image and the original images.

5.4.2 2 Testing Models 3a, 3b, and 3c and model 4a

For the testing of models 3a, 3b, 3c and 4a, tables were produced showing their classification results, highlighting correct prediction (see figure 1 below).

From the results it becomes quite clear that the model is performing worse than with the data from the C. Kampouris et al (2016).

However, as may noticed notice apart from the camera images 6 and 7 classified using 3a every other instance where images 6 and 7 where classified resulted in a correct prediction, which could indicate based on the images viewed in “homemade images”, that the model does identify colour as a distinguishing feature of the fabrics, alternatively could a reason for this occurrence also be the size of the woven structures.

Based on this information it can be reasoned that if zoom was applied as an augmentation and an additional augmentation was applied to maximize colour diversity, the same models could have performed better.

Additional does Model 3c stand out as one of the most consistent classifiers when looking at the correct classifications where it predicted the same 3 images for phone and camera as cotton.

Whereas Model 3a performed the worst overall predicting only 4 images correctly, compared to the other models which all predicted 6 correctly, displaying that dropout and regularization layers did cause an improvement in the models’ ability to generalize.

Model-3a-Aug-fa.h5

ID	Device	Actual Type	Prediction
1	Camera	Cotton	Silk
2	Camera	Cotton	Polyester
3	Camera	Cotton	Wool
4	Camera	Cotton	Polyester
5	Camera	Cotton	Polyester
6	Camera	Cotton	Polyester
7	Camera	Cotton	Polyester
8	Camera	Cotton	Cotton
9	Camera	Cotton	Polyester
10	Camera	Cotton	Polyester
11	Camera	Cotton	Polyester
1	Phone	Cotton	Silk
2	Phone	Cotton	Silk
3	Phone	Cotton	Polyester
4	Phone	Cotton	Cotton
5	Phone	Cotton	Wool
6	Phone	Cotton	Cotton
7	Phone	Cotton	Cotton
8	Phone	Cotton	Silk
9	Phone	Cotton	Silk
10	Phone	Cotton	Silk
11	Phone	Cotton	Silk

Model-3c-Aug-fa-lr=5e-4-ext.h5

ID	Device	Actual Type	Prediction
1	Camera	Cotton	Silk
2	Camera	Cotton	Silk
3	Camera	Cotton	Cotton
4	Camera	Cotton	Silk
5	Camera	Cotton	Silk
6	Camera	Cotton	Cotton
7	Camera	Cotton	Cotton
8	Camera	Cotton	Denim
9	Camera	Cotton	Polyester
10	Camera	Cotton	Polyester
11	Camera	Cotton	Wool
1	Phone	Cotton	Polyester
2	Phone	Cotton	Silk
3	Phone	Cotton	Cotton
4	Phone	Cotton	Silk
5	Phone	Cotton	Polyester
6	Phone	Cotton	Cotton
7	Phone	Cotton	Cotton
8	Phone	Cotton	Silk
9	Phone	Cotton	Polyester
10	Phone	Cotton	Silk
11	Phone	Cotton	Polyester

Model-3b-Aug-fa.h5

ID	Device	Actual Type	Prediction
1	Camera	Cotton	Polyester
2	Camera	Cotton	Polyester
3	Camera	Cotton	Silk
4	Camera	Cotton	Polyester
5	Camera	Cotton	Silk
6	Camera	Cotton	Cotton
7	Camera	Cotton	Cotton
8	Camera	Cotton	Cotton
9	Camera	Cotton	Polyester
10	Camera	Cotton	Wool
11	Camera	Cotton	Cotton
1	Phone	Cotton	Polyester
2	Phone	Cotton	Silk
3	Phone	Cotton	Polyester
4	Phone	Cotton	Nylon
5	Phone	Cotton	Denim
6	Phone	Cotton	Cotton
7	Phone	Cotton	Cotton
8	Phone	Cotton	Silk
9	Phone	Cotton	Polyester
10	Phone	Cotton	Polyester
11	Phone	Cotton	Polyester

Model-4a-Aug-fa.h5

ID	Device	Actual Type	Prediction
1	Camera	Cotton	Silk
2	Camera	Cotton	Polyester
3	Camera	Cotton	Polyester
4	Camera	Cotton	Polyester
5	Camera	Cotton	Cotton
6	Camera	Cotton	Cotton
7	Camera	Cotton	Cotton
8	Camera	Cotton	Cotton
9	Camera	Cotton	Silk
10	Camera	Cotton	Polyester
11	Camera	Cotton	Wool
1	Phone	Cotton	Polyester
2	Phone	Cotton	Silk
3	Phone	Cotton	Polyester
4	Phone	Cotton	Polyester
5	Phone	Cotton	Denim
6	Phone	Cotton	Cotton
7	Phone	Cotton	Cotton
8	Phone	Cotton	Silk
9	Phone	Cotton	Polyester
10	Phone	Cotton	Silk
11	Phone	Cotton	Polyester

Figure 1 Result tables for models 3a,3b,3c and 4a

5.5 Finalising implementation and design of the Executable

All improvements from Methods [4.5] have been implemented, however two additional sources other than James V. (2021) were required however to complete the explanations on how to identify fabrics which are Robert J. (2021) and the Sewport support team (2022).

See [Appendix \[8.3\]](#)

Chapter 6: Conclusion and Discussion

6.1 Objectives

This section will address the projects' objectives, how and for what reason they changed compared to the PDD or were further detailed, and the extent to which they have been fulfilled.

6.1.1 Change to initial objectives

Aside from changes to the objectives that focus on clarifying the objective itself, during the course of the project one objective was changed from the one of original objectives that is stated in the PDD document.

"1. The application will be an executable that is not reliant on me for the provision of external resources such as hosting a server (the software operates offline once downloaded)."

This was changed to:

"1. The final classification model will be accessible through a hosted service or standalone application, that will be provided through a desktop, mobile or web application that will allow any user to load an image of their choice, which will be classified using the model and have the result returned to the user. The application should be easy to operate by keeping the amount of user required actions to a minimum."

The reason for this change is that after the development of the web application the realisation was made that with the existence of the web application there is no reason for the existence for an additional local service, given that the main objective for the development of an application is to improve accessibility which is better achieved by the web application alone as it provides access to the service to a multitude of devices, only requiring them to have access to the internet and the ability to store an image for upload.

6.1.2 Objectives met

The primary objective as stated in [Introduction \[1.2\]](#) of achieving a model accuracy of 70% on average has been achieved for a collection of 6 different fabric types, which are, cotton, denim, nylon, polyester, silk and wool with the least represented fabric reaching an accuracy of ~50% which are wool, nylon and silk and the remaining fabrics reaching an accuracy of above 70%, this, allows for the conclusion that the model is able to perform well given enough data where the threshold for the minimum amount of data required lies at roughly 600 images, given the results in [Results \[5.2.3.2\] figure 2](#) where denim with that amount of samples is correctly classified 75% of the time.

For the additional objectives, a table was prepared below to showcase how these were achieved.

Objective Number	Additional Objectives
1	Developed a web application that allows for the described functionality providing quick and easy access through a web page to load an image taken by the user or another source, which results in the image to be uploaded to the server for classification and returning the results of the classification, being the fabric identified and the predictive power of the classification result. Which also implements client-side file validation and server side filename sanitization and concurrency measures in the form of a universally unique identifier (uuid) to which a filename is change when temporarily stored on the system.
2	Sadly due problems encountered model adjustments could not be performed as effectively as possible limiting the adjustments to changes in model inputs size, adding dropout layers and regularization layers, properly documented and explaining the working performed to reach each stage, additionally while not included in the report

	there exists early working attempt which diverged from the planned process due to which it was abandoned, however not before producing a model which did achieve a near 80% accuracy in just 50 epochs. This model was included for the final training stage in Results [5.4] , however the best performing model was created in the process documented in Results [5.2] . Which gave important insights to the final output of this project for real world fabric classification.
3	The adjustment tested for the duration of this project due to major delays was rotation, which by itself proved as an essential factor in the performance of models trained using the fabric datasets and showed how import augmentation and in the future an expansion in the number samples in the original “Fabrics” dataset would substantially improve the possible improvements to be made in this problem domain of computer vision. Additionally, did Results [5.4] indicate that the fabrics dataset lacks diversity in zoom/distance range and colour diversity, such that the models produced showed bias for the classification of cotton in light grey fabrics.

From the additional objectives only objective “b” was reached where there was additional information provided on the webpage describe additional traits to identify fabrics by.

For objective “c” a conclusion was reached that there no universal method to take images by camera as not all cameras are as advance as other and allow operation such as the change of the F value which affects image focus, which came to the authors realization after producing their own set of images using a phone and a normal camera which could be viewed as a sub objective of objective “a”.

How code reuse affected the project

In this project code from third parties was reused where mentioned or explicitly stated by the author throughout this document, however the only code reuse with a substantial impact on the project was a tutorial used on tensorflow.org which produced the 4-leayer model in [Results \[5.2.2\]](#) as it is the project starting point much of the other models can be viewed as variations of the same models, however, the author still ensured to test, document and analyse the outputs produced for each model and its variations to ensure that the work performed outweighs the initial work load taken away by creating a model with help from a tutorial and use it for their own project, which the author believes has been achieved.

Problems encountered

Various problems where encountered which caused major delays and hindered the project progress

- The start of the first training cycle above 50 epochs was delayed until 4 weeks into the project due to a lack of gpu acceleration available to the author, this was resolved by purchasing a google collab subscription which offered gpu acceleration and already came prepared with TensorFlow. In the time until the was resolve the author attempted training on model 2 where 50 epochs required 1h+ to complete meanwhile collab required only 5 minutes. However, as the author was reluctant to pay to be able to reasonably complete the project it first was attempted to configure gpu acceleration which they later learned was not possible on their machine. After this the author contacted city, university of London about their own cloud-based services to access additional computation power, in case if it had been necessary during a later stage of the project, at that point in time.
- While collab solved the problem of training speed, as the service was staying connected to the collab instance, connection issues could cause an entire training attempt to be wasted,
- For methods [Methods \[4.3\]](#) it was initially stated:
“Due to the delays encountered as mentioned in the previous section, the author limited the time spent in this section to create a satisfying model with the optimization techniques already explored and the possible addition of new hidden layers”

This led to the creation of a set of model 3 V5 (now called model 4a), which was decided to not be used, due to the numbers of adjustments untested and that training for models 3 will be restarted in addition to a change in the naming convention used and creation of the utility file "my_utils.py".

- Finally, the authors' private life was very busy throughout the project with other assignments and added stress of looking for a job to work after graduation having to practice for and partake in interviews.

Conclusion

The project is considered a success despite the many issues encountered, nonetheless it is regrettable for the author that not as much testing for different model adjustments and augmentations, the addition of noise or transfer learning with the VGG-16 model architecture that was applied in the project by Yasith Sanura Perera (2020), additionally it is unfortunate for the author to make a possibly major insight right before the submission deadline of the project in the form of [Results \[5.4\]](#).

However, the author believes that a lot can be learned from the mistakes made such as, underestimating the computational power required for CNN models, and not having researched appropriate naming conventions and file organizational structures.

Chapter 7: Reference List

Byju's Gate. Difference Between Machine Learning and Deep Learning. Byju's (2022). Available at: <<https://byjus.com/gate/difference-between-machine-learning-and-deep-learning/>> (Accessed: 15.02.2022)

LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* 521, 436–444 (2015). Available at: <<https://doi.org/10.1038/nature14539>> (Accessed: 15.02.2022)

CodeBullet. What are Neural Networks || How AIs think. Youtube (2018). Available at: <<https://www.youtube.com/watch?v=JeVDjExBf7Y>> (Accessed: 15.02.2022)

Yasith Sanura Perara. AI for Textiles – Convolutional Neural Networks Based Fabric Structure Classifier. Towardsdatascience (2020). Available at: <<https://towardsdatascience.com/ai-for-textiles-convolutional-neural-network-based-fabric-structure-classifier-c0db5433501d>> (Accessed: 16.02.2022)

C. Kampouris, S. Zafeiriou, A. Ghosh, S. Malassiotis. The Fabrics Dataset. ibug.doc (2016). Available at: <<https://ibug.doc.ic.ac.uk/resources/fabrics/>> (Accessed: 14.02.2022)

C. Kampouris, S. Zafeiriou, A. Ghosh, S. Malassiotis. Fine-grained material classification using micro-geometry and reflectance, 14th European Conference on Computer Vision, Amsterdam (2016). (Accessed: 14.02.2022)

Arun Gandhi. Data Augmentation | How to use Deep Learning when you have Limited Data — Part 2. Nanonets Blog (2021). Available at: <<https://nanonets.com/blog/data-augmentation-how-to-use-deep-learning-when-you-have-limited-data-part-2/>> (Accessed: 18.02.2022)

James V., How To Tell If Fabric Is... (How to Identify Fabrics Easily). Sewingiscool (2021). Available at: <<https://sewingiscool.com/how-to-identify-fabrics/>> (Accessed: 14.02.2022)

Kurtis Pykes. Fighting overfitting with L1 or L2 regularisation. Neptune Blog (2021). Available at: <<https://neptune.ai/blog/fighting-overfitting-with-l1-or-l2-regularization>> (Accessed: 02.04.2022)

Frank Andrade. How to Easily Build Your First Machine Learning Web App in Python. Towards Data Science (2022). Available at: <<https://towardsdatascience.com/how-to-easily-build-your-first-machine-learning-web-app-in-python-c3d6c0f0a01c>> (Accessed: 07.04.2022)

Faizan Amin. Deploy keras image classification model using flask and Docker on Azure. ML Hive (2021). Available at: <<https://www.mlhive.com/2021/01/deploy-keras-image-classification-model-using-flask-and-docker-on-azure>> (Accessed: 07.04.2022)

Robert Jameson. how-to-tell-if-fabric-is-100-cotton. fabricfits (2021). Available at: <<https://www.fabricfits.com/how-to-tell-if-fabric-is-100-cotton>> (Accessed: 10.05.2022)

Sewport support team. What is Denim Fabric: Properties, How its Made and Where. Sewport (2022). Available at: <<https://www.sewport.com/fabrics-directory/denim-fabric>> (Accessed: 10.05.2022)

Chapter 8: Appendices

8.Appendix A: Project Definition Document

Project Definition Document
*For the BSc computer science Final year project
at City, University of London
Academic year 2021/2022*

Project title:

Visual Fabric Classification using Neural Networks

Student: Nathan Odibo | Email: nathan.odibo@city.ac.uk

Consultant: Dr. Martin Walter

Proposed by: Nathan Odibo

Arrangements for proprietary interests: None

Any other promises you are making in order to secure acceptance of the project: None

Word count: 1255 (Excluding Cover sheet, Index and Appendixes, Including references and section headers)

INDEX

Project Definition Document.....	45
Proposal.....	47
Problem to be solved.....	47
Project Objectives.....	48
Project Beneficiaries.....	48
Work Plan	48
Project Risks.....	49
Risks to your project.....	49
Risks that your project poses to others.....	50
References:.....	50
Appendix: Work Plan (Gant Chart)	51
Appendix: Research Ethics Review Form.....	52

Proposal

Problem to be solved

In today's time, we have access to a multitude of different fabrics which all have their unique traits such as their feel to the touch, their ability to absorb or repel water, the friction it produces, and how much wear and tear it can withstand.

So, it is all the more important to choose the correct fabric or combination of different layers of fabrics to make clothes to fulfill a certain need.

Nonetheless, it is not quite easy to identify a fabric based on touch alone for a common individual, especially if they never used that fabric before, and the most proven method of identifying a fabric "the burn test" is not a reliable way for everyone to identify a fabric and not everyone might "read" the burning of the fabric correctly, besides that not all fabrics burn differently which makes it even harder to apply. (James V. 2021)

Among those problems, a common individual may not be too eager to learn about all the ways to identify all of the different kinds of fabrics and remember each distinguishing feature to identify if it is the fabric they are looking for, including the way it behaves when burning.

On top of that, they may not even have the option of burning a piece of fabric since the flames are a risk to their surroundings, and the fabric they want to test might be too scarce and/or expensive to cut a piece out of.

The project aims to solve all these issues by providing a software solution to identify fabrics based on an image taken of the fabric based on the fabrics available in a dataset that will be used to train an AI model which will be trained using different classification models to choose the one with the best accuracy to display to an end user of the software solution what kind of fabric they have.

The data set used will be one provided on www.ibug.doc.ic.ac.uk/resources/fabrics/ by C. Kampouri et al. (2016) that was made for a research paper by the same author group.

Project Objectives

The project's main objective is to create a classification model using neural networks for image classification, with an average accuracy of at least 70% as the dataset used does have a lack of certain images of some fabric types for the model to be trained on (e.g. There are no images for "fur" to train the model on at this point) and 70% is the average accuracy obtained within the report of C. Kampouri et al. (2016) which focuses on fabric classification.

Additional Objectives:

4. The application will be an executable that is not reliant on me for the provision of external resources such as hosting a server (the software operates offline once downloaded).
5. Different models and model options (e.g. adjusting layers and the amount thereof and changing the activation function) will be used to identify the model with the best accuracy for the classifications.
6. Artificially increase the dataset size by translating images (rotate, scale, add a colour layer), if there is only a small collection of certain images in the dataset for certain fabrics, and test the effects on the well-represented data.
7. The program will take an image and return the output from the classification with the confidence value and name of the detected fabric.

Optional Objectives:

- Improve the dataset by gathering additional images for fabrics that lack many/any image sets (e.g. collect images for "fur") and retrain the model to include them
- Make the software accessible through mobile devices or the web.
- Include more information about the fabric detected, such as additional checks that can be performed through touch or "the burn test".
- Identify the conditions in which photos taken will yield the best results (distance, lighting, image specifications)

Project Beneficiaries

On the success of the project, hobby sewers and commercial sewers alike should be able to benefit from it, should they be unsure about the type of a piece of fabric or don't have enough experience with the fabric to identify it by themselves.

The research could also benefit another researcher or larger companies who are interested in expanding the research in the classification of fabrics or automation of certain processes which would require a fabric classification model.

Work Plan

This project will mainly python the panda library for data preparation and the TensorFlow, Keras library to perform the classification to and to create the predictions
other libraries may be used if required, but these are the main ones, but tools such as anaconda which in addition to TensorFlow and Keras allow me to use Spyder and Jupiter-notebook to easily interpret the model's outputs. (Gant chart in appendix)

Project Risks

Risks to your project

No.	Risk	Likelihood	Severity	Prevention/Mitigation
1.	Low classification accuracy for underrepresented fabrics.	7	4	Identify the threshold of data required to perform accurate predictions, reduce the range of fabrics that will be classified. Artificially increase data through different transformations. Collect more data myself using my own camera.
2.	2 or more fabrics are commonly mistaken.	4	4	Artificially expand the dataset to allow the model to extract new features or to revalue the currently identified features. Add a clause within the program to display the top k number of fabrics if they share a similar likelihood of being correct. Analyze such occurrences beforehand to indicate if a fabric is similar to other fabrics when it has identified a fabric with this issue.
3.	Unable to artificially expand the data may reduce the accuracy of the classifier. By making the classifier disregard important features in the data.	3	5	Flipping of images should have no negative impact on the data for this problem, scaling and cropping should not have a negative impact to a degree, rotation, and translation are options for data augmentation, but they will create leftover space that will have to be accounted for in some way. Noise might be used to simulate worse camera quality.
4.	The camera quality might not be high enough	5	4	My phone camera has a good quality for the purpose of data collection. And alternatively, I should be able to either borrow a camera better suited for the task.
5.	Lack of access to fabrics to photograph.	1	3	The same as the images in the dataset, I can take photographs of clothes of the looked-for fabric at clothing shops. Alternatively, there are many more datasets or images of fabrics located on the web for free use.

6.	Loss of my work by breaking/loss/theft of my computer.	1	10	Create a git repository on which my work will be maintained.
7.	Unexpected delays	7	7	Schedule enough time within my workplan, to allow delays without threatening to be late for the final submission deadline.

Risks that your project poses to others

To avoid users of the software making a misinformed choice, any users will always be shown the confidence of accuracy of the prediction to indicate that the software is not flawless and to avoid that a user makes a choice based solely on the outcome of any first prediction the software makes.

References:

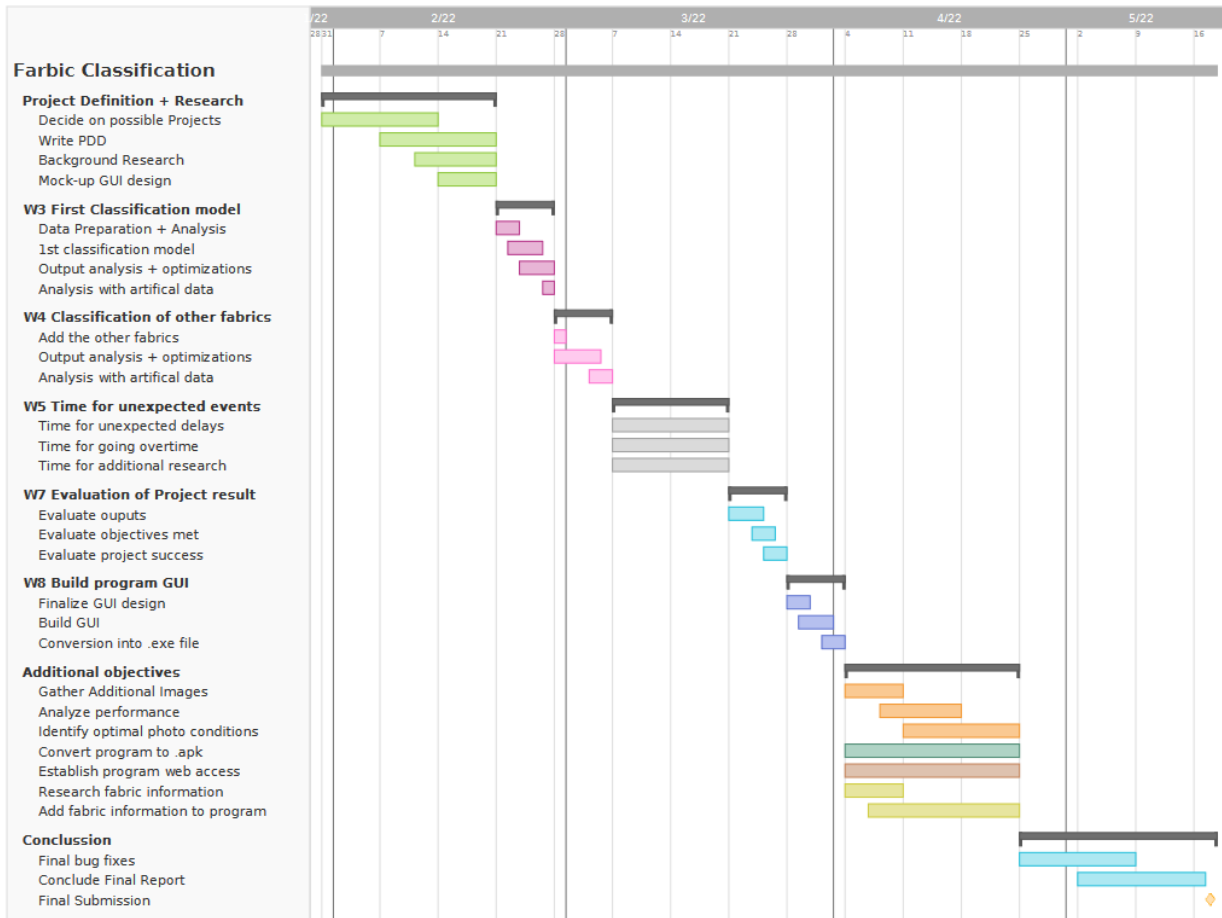
James V., 2021. How To Tell If Fabric Is... (How to Identify Fabrics Easily). [online] Sewingiscool. Available at: <<https://sewingiscool.com/how-to-identify-fabrics/>>(Accessed: 14.02.2022)

C. Kampouris, S. Zafeiriou, A. Ghosh, S. Malassiotis. (2016) The Fabrics Dataset. [online] ibug.doc. Available at: <<https://ibug.doc.ic.ac.uk/resources/fabrics/>>(Accessed: 14.02.2022)

C. Kampouris, S. Zafeiriou, A. Ghosh, S. Malassiotis. (2016) Fine-grained material classification using micro-geometry and reflectance, 14th European Conference on Computer Vision, Amsterdam.

Appendix: Work Plan (Gant Chart)

teamgantt
Created with Free Edition



Appendix: Research Ethics Review Form

Research Ethics Review Form: BSc, MSc and MA Projects

Computer Science Research Ethics Committee (CSREC)

<http://www.city.ac.uk/departments-computer-science/research-ethics>

Undergraduate and postgraduate students undertaking their final project in the Department of Computer Science are required to consider the ethics of their project work and to ensure that it complies with research ethics guidelines. In some cases, a project will need approval from an ethics committee before it can proceed. Usually, but not always, this will be because the student is involving other people ("participants") in the project.

In order to ensure that appropriate consideration is given to ethical issues, all students must complete this form and attach it to their project proposal document. There are two parts:

PART A: Ethics Checklist. All students must complete this part. The checklist identifies whether the project requires ethical approval and, if so, where to apply for approval.

PART B: Ethics Proportionate Review Form. Students who have answered "no" to all questions in A1, A2 and A3 and "yes" to question 4 in A4 in the ethics checklist must complete this part. The project supervisor has delegated authority to provide approval in such cases that are considered to involve MINIMAL risk. The approval may be **provisional** – *identifying the planned research as likely to involve MINIMAL RISK*. In such cases you must additionally seek **full approval** from the supervisor as the project progresses and details are established. **Full approval** must be acquired in writing, before beginning the planned research.

A.1 If you answer YES to any of the questions in this block, you must apply to an appropriate external ethics committee for approval and log this approval as an External Application through Research Ethics Online - https://ethics.city.ac.uk/		<i>Delete as appropriate</i>
1.1	Does your research require approval from the National Research Ethics Service (NRES)?	NO
1.2	Will you recruit participants who fall under the auspices of the Mental Capacity Act?	NO
1.3	Will you recruit any participants who are currently under the auspices of the Criminal Justice System, for example, but not limited to, people on remand, prisoners and those on probation?	NO
A.2 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee, you must apply for approval from the Senate Research Ethics Committee (SREC) through Research Ethics Online - https://ethics.city.ac.uk/		<i>Delete as appropriate</i>
2.1	Does your research involve participants who are unable to give informed consent?	NO
2.2	Is there a risk that your research might lead to disclosures from participants concerning their involvement in illegal activities?	NO

2.3	Is there a risk that obscene and or illegal material may need to be accessed for your research study (including online content and other material)?	NO
2.4	Does your project involve participants disclosing information about special category or sensitive subjects?	NO
2.5	Does your research involve you travelling to another country outside of the UK, where the Foreign & Commonwealth Office has issued a travel warning that affects the area in which you will study?	NO
2.6	Does your research involve invasive or intrusive procedures?	NO
2.7	Does your research involve animals?	NO
2.8	Does your research involve the administration of drugs, placebos or other substances to study participants?	NO
<p>A.3 If you answer YES to any of the questions in this block, then unless you are applying to an external ethics committee or the SREC, you must apply for approval from the Computer Science Research Ethics Committee (CSREC) through Research Ethics Online - https://ethics.city.ac.uk/</p> <p>Depending on the level of risk associated with your application, it may be referred to the Senate Research Ethics Committee.</p>		<i>Delete as appropriate</i>
3.1	Does your research involve participants who are under the age of 18?	NO
3.2	Does your research involve adults who are vulnerable because of their social, psychological or medical circumstances (vulnerable adults)?	NO
3.3	Are participants recruited because they are staff or students of City, University of London?	NO
3.4	Does your research involve intentional deception of participants?	NO
3.5	Does your research involve participants taking part without their informed consent?	NO
3.5	Is the risk posed to participants greater than that in normal working life?	NO
3.7	Is the risk posed to you, the researcher(s), greater than that in normal working life?	NO
<p>A.4 If you answer YES to the following question and your answers to all other questions in sections A1, A2 and A3 are NO, then your project is deemed to be of MINIMAL RISK.</p> <p>If this is the case, then you can apply for approval through your supervisor under PROPORTIONATE REVIEW. You do so by completing PART B of this form.</p> <p>If you have answered NO to all questions on this form, then your project does not require ethical approval. You should submit and retain this form as evidence of this.</p>		<i>Delete as appropriate</i>
4	Does your project involve human participants or their identifiable personal data?	NO

*****If these items are not available at the time of submitting your project proposal, then provisional approval can still be given, under the condition that you must submit the final versions of all items to your supervisor for approval at a later date. All such items must be seen and approved by your supervisor before the activity for which they are needed begins. Written evidence of final approval of your planned activity must be acquired from your supervisor before you commence.***

Changes

If your plans change and any aspects of your research that are documented in the approval process change as a consequence, then any approval acquired is invalid. If issues addressed in Part A (the checklist) are affected, then you must complete the approval process again and establish the kind of approval that is required. If issues addressed in Part B are affected, then you must forward updated documentation to your supervisor and have received written confirmation of approval of the revised activity before proceeding.

Templates for Consent and Information

You must use the templates provided by the University as the basis for your participant information sheets and consent forms. You **must** adapt them according to the needs of your project before you submit them for consideration.

Participant Information Sheets, Consent Forms and Protocols must be consistent. Please ensure that this is the case prior to seeking approval. Failure to do so will slow down the approval process.

We strongly recommend using Qualtrics to produce digital information sheets and consent forms.

Further Information

<http://www.city.ac.uk/departments-computer-science/research-ethics>

<https://www.city.ac.uk/research/ethics/how-to-apply/participant-recruitment>

<https://www.city.ac.uk/research/ethics>

8. Appendix B: Reuse Summary

As by the “BSc MSci_Proj_Handbook_2021-22_uploaded_22_03_2022” the author adhered to section

“8 Good Professional Practice” outlining in methods and results the occurrences where code (where all runnable and used code is located in the authors GitHub repository at: https://github.com/naod488/Fabric_Classification) was reused or (at least added a comment among the code the location from where code was taken from) and the impact of its reuse on the project, these occurrences are:

- The ipynb files of which followed a tutorial (https://www.tensorflow.org/tutorials/load_data/images) from which the model was further adapted surrounding code above the line of code which produces the model summary is included in that set with the exception of the 3 Data_Analysis(...).ipynb files where plots and file counts were created by me.
- “my_utils.py” located in the authors GitHub repository, contains re-use outside of the tensorflow.org tutorials, which is a method in a keras.io (https://keras.io/examples/vision/image_classification_from_scratch/) tutorial and one case where code of one of the already referenced Literature source Yasith Sanura Perera (2020), was re used, which in that case merely highlights the re-use of the as the code added is the code which adds a checkpoint to the model, and doesn’t allow much variation.
- On the authors github repository the python scripts which load the model, both “fabric_app.py” files and “app.py” located in “fabric_app”-> “js” and “tkinter” and “py_server” respectively, reuse a repeated piece of code used already in the ipynb files and notebook files from a tensorflow.org tutorial (https://www.tensorflow.org/tutorials/load_data/images)
- Additionally with the authors github repository in “fabric_app”-> “py_server”-> “app.py” code was adapted from multiple sources as indicated by comments and combined with code produced by the author to provide new solutions.
- For the same folder reference previously, text was inserted from additional literature to add to page content, thus technically code, it is a literature reference

8.1 Source Code and instructions for use of notebook files and web app

All source code is available at: https://github.com/naod488/Fabric_Classification

All corresponding saved h5 files uploaded to:

Google Drive:

<https://drive.google.com/drive/folders/1afrvWWTIm5FQYpH1wWQzKZMVMjSxsGtx>

OneDrive:

https://cityuni-my.sharepoint.com/personal/nathan_odibo_city_ac_uk/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fnathan%5Fodibo%5Fcity%5Fac%5Fuk%2FDocuments%2FFabric%5FClassification

Instructions to start the localhost python flask webapp (GitHub: README.md):

"""

Fabric_Classification

A Convolutional Neural Network (CNN) fabric classification model creation and application deployment project

This folder contains various ipynb files setup for a google collab environment, h5 files for trained models, some utility files, and a folder containing various

implementations for a tkinter, node.js and python flask for a CNN image classification application.

Model creation notebook ipynb files, trained models saves h5 files and datasets

The project consists of several 4 layer models that classifies between 2 or 6 images of depending on the Model version

Models version 1 a 2 layer model to classify image of Denim and Wool

Models version 2 is a 4 layer model which also classifies images of Denim and Wool

Models version 3 is a 4 layer model which classifies images of Cotton, Denim, Nylon, Polyester, Silk and Wool

Models version 4 is a 6 layer model which classifies images of Cotton, Denim, Nylon, Polyester, Silk and Wool

The datasets are:

fabrics_adjusted: A dataset consisting of fabric types Cotton (), Denim (), Nylon (), Polyester (), Silk () and Wool (), selected from the original dataset (link below)

on the basis of the number of samples and omitting irrelevant categories

(used for models version 3 and 4)

fabrics_simple: A dataset consisting of fabric types Denim () and Wool (), from the fabrics_adjusted dataset on the basis of the number of samples

(used for models version 1 and 2)

The original complete fabrics dataset can be obtained from:

Fabrics: the original dataset is available at <https://ibug.doc.ic.ac.uk/resources/fabrics/>

Datasets for the ipynb files and h5 for the trained models in the corresponding ipynb files are available on the authors google drive:

<https://drive.google.com/drive/folders/1afrvWWTIm5FQYpH1wWQzKZMVMjSxsGtx?usp=sharing>

To Run a model for training

This section assumes system has been set up according to the Project Report chapter 4 Methods System Setup

Make sure you use the appropriate dataset the model is ment for as explained in the section above

- 1. ensure my_utils.py is located in the same folder as the to run ipynb notebook file*
- 2. remove the first 1 code block if runing on a local machine (remove the code to mount the google drive*
- 3. update the file path variables to mach your environment*

(pathnames to update: variable "path" after the import in teh 3rd/2nd code block

parameters for "checkpoint = my_utils.generate_callback(...,....)" after model.comple()

!Additional notice!

when a model is trained the checkpoint will overwritte any file with the same path/name in the user directory

The CNN application (located in fabric_app)

The completed application in the fabric app folder is the python server implementation using flask,

Notes: a h5 file has already been located in the same folder as the executable

it may be replace with any other h5 files but the variable for its path "MODEL" will be required to be changed appropriately,

addtionally, if it was changed to a h5 file for the "fabrics_simple"

dataset requires a change in the list variable called "TYPES"

to run:

- pip install, flask and tensorflow (or follow instructions in the Project Report Chapter 4 Methods System Setup)

- exectue the app.py file and wait for the console window to show "Running on "http://127.0.0.1:5000/"

- open the webbrowser and navigate to the "http://127.0.0.1:5000/"

from this point on the user will be shown a page with the option to upload an image, if done so the user will be redirected to the results page, showing the outcome of the classification.

The image uploaded will be sorted on the hosting machine temporarily using a universally unique identifier to generate a unique filename to avoid concurrency issues

when multiple users upload an image with possibly the same name at the same time.

Additional Notes:

3 Test files have been provided in "fabric_app" to submit as images:

- test_giberish.png should result in the user being notified of that it wasn't possible to classify the image

- test_giberish.txt should result in an alert to be displayed, if a user where to disable javascript and somehow upload the txt

or any other non image file of their choosong, the user will be notified that it wasn't possible to classify the image

- test_Wool_ima_1.png (the fabrics dataset "Wool_ima_1.png"), the user will be presented with a classifcation result, in case of the

original model configured it will present wool as the identified fabric.

Limitations:

- The classification result does not have the option to label the image uploaded as none of the fabric types.*
- The image to be classified is expected to possess a resolution of 400x400 for larger images, will be scaled down for evaluation,*
for that reason the image needs to be of a very high quality and close up, and possibly cropped.

There exist two alternate implementations, one using python tkinter and the other using node js, these implementations are functionally complete in terms of allowing an image to be loaded and classified.

To run:

- Tkinter: execute "fabric_app.py" libraries required view imports at the top of file source*
- Node.js: execute the "node ." command within the folder after you see the host and port number displayed*
navigate to localhost:3000 (default configuration)

Utility files

Dataset_Analysis('fabrics_...').ipynb:

Loads the images and displays a figure for the number of samples in each category by detecting the number of png files in each subfolder

my_utils.py:

a file with helper functions for the model creation notebook files, which is expected to be located with the same folder as the notebook files themselves

Fair use

This project falls under fair use, results in the form of the fabrics dataset and datasets derived from this dataset and the trained model h5 files

are restricted for non-commercial use as by the author of the original dataset C. Kampouris et. al (2016)

References

C. Kampouris, S. Zafeiriou, A. Ghosh, S. Malassiotis. *The Fabrics Dataset*. *ibug.doc* (2016).

Available at: <<https://ibug.doc.ic.ac.uk/resources/fabrics/>> (Accessed: 14.02.2022)

""

8.2 Notebook files model related diagrams

All notebook files are located in the authors github

at: https://github.com/naod488/Fabric_Classification to ease viewing of model structures and plots.

All notebook files were not cleared to keep test and result in tact to analysis these without requiring the training of an entire model, however in some cases the model has rerun by accident so not all models contain the exact figures showcased in [Results \[5.2\]](#).

8.3 Website pages

Actual webapp located at:

https://github.com/naod488/Fabric_Classification/tree/main/fabric_app/py_server

For execution consult the readme in Appendix [8.1] or on:

https://github.com/naod488/Fabric_Classification

Fabric Classifier

This website is the image classifier produced by Nathan Odibo for the dissertation project at City, University of London.

The classifier uses a convolutional neural network for the purposes of image classification. It possess the ability to distinguish among 6 fabrics [Cotton, Denim, Nylon, Polyester, Silk, Wool].

To record and upload your image ensure that the image is close up, as sharp as possible and of 400x400 resolution, or cropped down to the specified resolution from a larger image, it should be taken looking straight at the fabric with no direct shadow or light obscuring the image. For examples on how images should look like visit www.ibug.doc.ic.ac.uk/resources/fabrics/.

Limitations:
The default model configured for use for this web application is a convolutional neural network that has achieved an over 75 percent accuracy on the validation split of the dataset it was trained on, provided on www.ibug.doc.ic.ac.uk/resources/fabrics/, as such the models shows to the environment in which the original data is taken in an struggles to perform well on custom data showing an accuracy of ~25 on a set of 22 images of cotton fabrics, 11 taken by a Sony camera and phone each. Therefore major caution is advised when relying on the results provided by this module on custom data.

Operation:

- Use the file selector below to select a .jpg or .png (recommended) image on your device.
- Press the upload button.
- If no errors occurred you will be shown the classification results for the uploaded image in the form of a % which indicates, of the identifiable fabrics, which image has the largest likelihood to be of a certain fabric.
- In addition to that a short summary will be provided on how you can perform a test to be certain that the fabric was correctly identified.

Upload an image for classification

Durchsuchen...

Keine Datei ausgewählt.

Daten absenden

By Nathan Odibo [visit the GitHub for the website source files](#)

Figure 1 Webapp main page

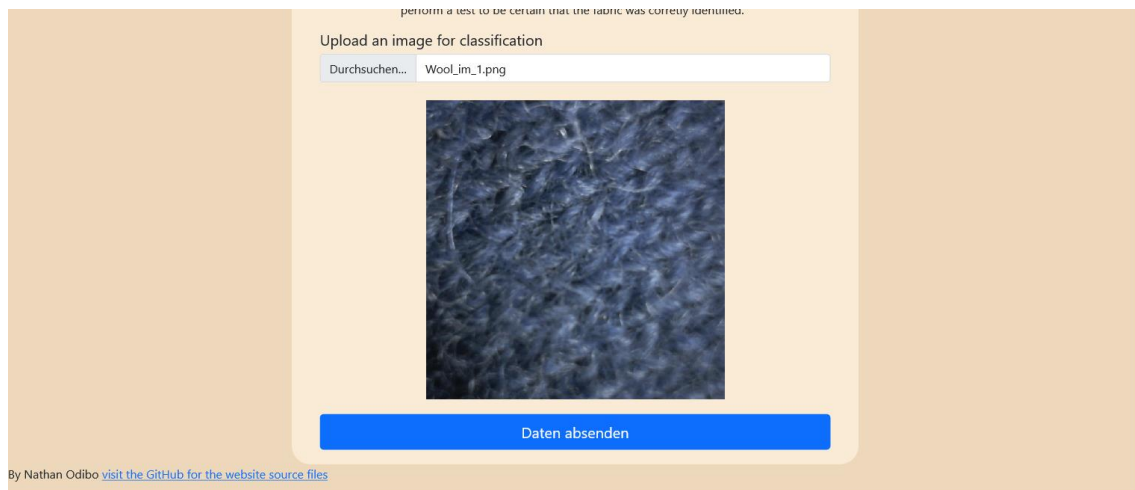


Figure 2 Webapp image selected for submission



Figure 3 WebApp image submitted classification results and additional fabric information text displayed



Figure 4 WebApp corrupted image selected after from classification result page



Figure 5 WebApp corrupted image submitted, error notification returned

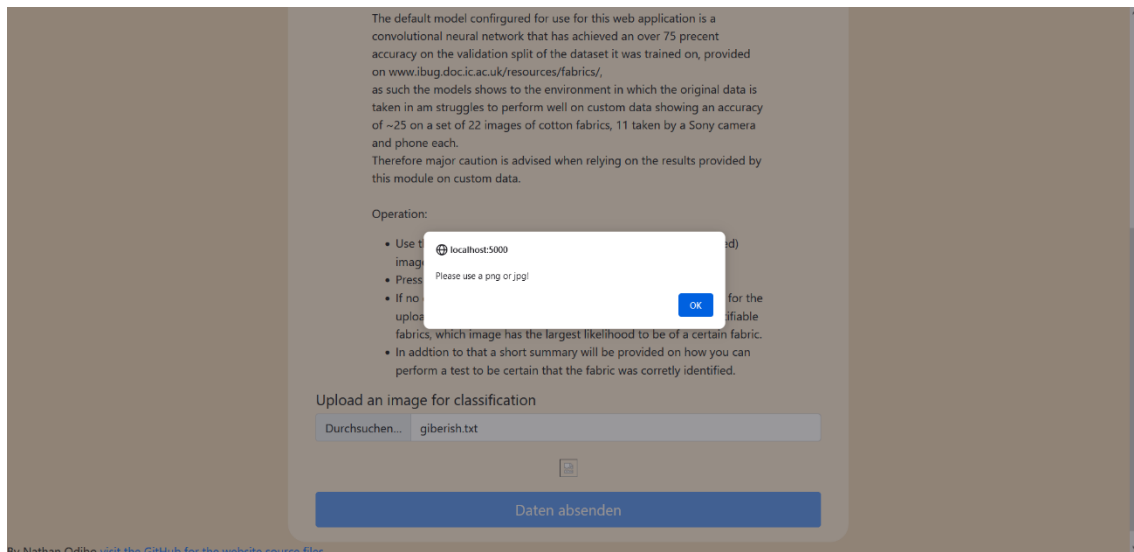


Figure 6 WebApp Invalid file type selected

8.4 Datasets and image snippets produced

Images such as datasets generated and including those used in the main body of the report were uploaded to the authors OneDrive:

https://cityuni-my.sharepoint.com/personal/nathan_odibo_city_ac_uk/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fnathan%5Fodibo%5Fcity%5Fac%5Fuk%2FDocuments%2FFabric%5FClassification